UNIVERSITY OF CALIFORNIA
RIVERSIDE

Calibration of Multi-LIDAR Systems for Real-Time Surface Management: Application in Bucket Wheel Reclaimer

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Mohammad S. Billah

September 2019

Dissertation Committee:

Dr. Jay A. Farrell, Chairperson
Dr. Matthew J. Barth
Dr. Amit K. Roy-Chowdhury

The Dissertation of Mohammad S. Billah is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

First and foremost, I would like to express my sincere thanks and gratitude to my advisor, Professor Jay A. Farrell for his guidance, support, and patience during my journey towards a Ph.D. His rigorous thinking and writing style has taught me how important it is for an engineer which is going to have a life-long influence on me. I feel extremely lucky to have been a part of his group.

Besides my advisor, I would like to thank the rest of my thesis committee: Professor Matthew Barth, and Professor Amit Roy-Chowdhury for their support. Professor Roy-Chowdhury is an excellent teacher and I learned a lot from his course. During my Ph.D., I had the opportunity to work with Professor Barth which has been an invaluable experience for me, and I cannot thank him enough for that. I would also like to thank Professor Anastasios Mourikis, he is a great teacher, the knowledge I gained from his courses were invaluable for me.

Completing this work would have been all the more difficult were it not for the support and friendship provided by the other members of the Control and Robotics Lab at UC Riverside. I would like to thank Dr. Paul Roysdon, Dr. Elahe Aghapour, Farzana Rahman, and Zeyi Jiang, for their support and insightful discussion. I also like to convey my special thanks to Arash Maskooki and Axel Barrau for the intellectual discussions we had in the lab.

Finally, I want to thank the support and encouragement from my parents. Without their help, I would not have been here. Most importantly, I want to thank my wife, Tahmida Mahmud, for her endless and unconditional love, encouragement and support.

To my parents and my wife for all the support.

ABSTRACT OF THE DISSERTATION

Calibration of Multi-LIDAR Systems for Real-Time Surface Management: Application in Bucket Wheel Reclaimer

by

Mohammad S. Billah

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, September 2019
Dr. Jay A. Farrell, Chairperson

Stockpile reclaiming using a Bucket Wheel Reclaimer (BWR) is an important part of stockyard management. The growth in demand for material handling over the years has drawn attention to improve the automation of the process. However, studies have shown that stockpiled products are being reclaimed at approximately 50% of their potential. This study focuses on the challenges in the automation of stockyard management system using a BWR.

For high accuracy point cloud computation and surface reconstruction of the stockpiled materials, accurate calibration is of crucial importance. This dissertation presents a calibration technique for multi-LIDAR systems to estimate the GNSS-LIDAR extrinsic parameters of BWR's. The approach presented works with one or more 2D LIDARs and does not require special markers (e.g., reflective tape) or surveyed locations (other than a DGNSS base station antenna). The method and its accuracy have been demonstrated using experimental data from a stockyard environment.

Regarding real-time management and control, the dissertation presents a technique

for real-time point cloud management, visualization, and feature extraction for large scale stockyards environments. The software solution described continuously manages the point cloud in real-time as the sensors stream data. It also displays the current stockpile on an interactive interface that allows the user to see the surface from different view-points, interrogate the coordinates of any surface location, and computes the BWR entry and exit point for automated operation. The software is tested using experimental data from a port located in Yantai, China.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation focuses on the calibration of multi-LIDAR systems and its application in the automation of a Bucket Wheel Reclaimer (BWR). BWR's are used to recover stockpile using its rotating wheel with buckets mounted. Automation of BWR reclaiming refers to the process of reclaiming all the stockpiled materials without human intervention. This is done by a iterative process where in each iteration, the entry and exit points of the bucket wheel for the next sweep are automatically computed and sent to the device controlling the motion of the BWR. The problem can be divided into three subproblems: calibration, surface reconstruction and control of Bucket Wheel Reclaimers (BWRs).

1.1 Calibration

Calibration refers to the process of estimating the rigid body transformation between the LIDAR and GNSS sensors mounted on the BWR. Accurate estimate of calibration parameters is essential for accurate point cloud generation and feature extraction.

1.2 Surface Reconstruction

Surface reconstruction refers to the process where raw LIDAR data and GPS position estimates are used to create a continuous representation of the stockpile in real-time. A triangulated irregular network (TIN) is used to represent the surface. Efficient and accurate point cloud data management and triangle update are required for precise and real-time surface reconstruction.

1.3 Control

Control refers to the process where features are extracted from the reconstructed surface and sent to the PLC (Programmable Logic Controller) to control the motion of the BWR for efficient and automated operation.

1.4 Main Contributions

The main contributions of the dissertation are

1. Development and implementation of a novel multi-LIDAR extrinsic calibration technique for GNNS-LIDAR system.

2. Implementation of an efficient spatial grid partitioning technique for point cloud management and a feature extraction method for optimal digging point selection.

3. Development and implementation of a real-time software for automatic operation of the BWR.

Chapter 2

Bucket Wheel Reclaimer

A BWR's function is to recover bulk material such as ores and cereals from a stockpile. They are used worldwide. A BWR is shown in Fig. 2.1. The *base* of the BWR travels along a *rail* to maneuver along and between stockpiles in the stockyard. Attached to the base is a *boom* that has two degrees-of-freedom, being able to change its pitch ($\pm 10^{\circ}$) and yaw relative to the base. The roll angle of the boom relative to the base is fixed. A rotating wheel of buckets is attached to the end of the boom that is opposite from the base. The wheel can be moved in three directions: horizontally in the direction parallel to the rail by moving the base along the rail; vertically along an arc by luffing (i.e., changing the pitch of) the boom; and, horizontally along an arc by slewing (i.e., changing the yaw of) the boom.

Older BWR systems compute position information using an onboard encoder system with calibrated reset points [35]. Modern BWR systems use Differential Global Navigation Satellite Systems (DGNSS) for wheel position and attitude determination [12]. They

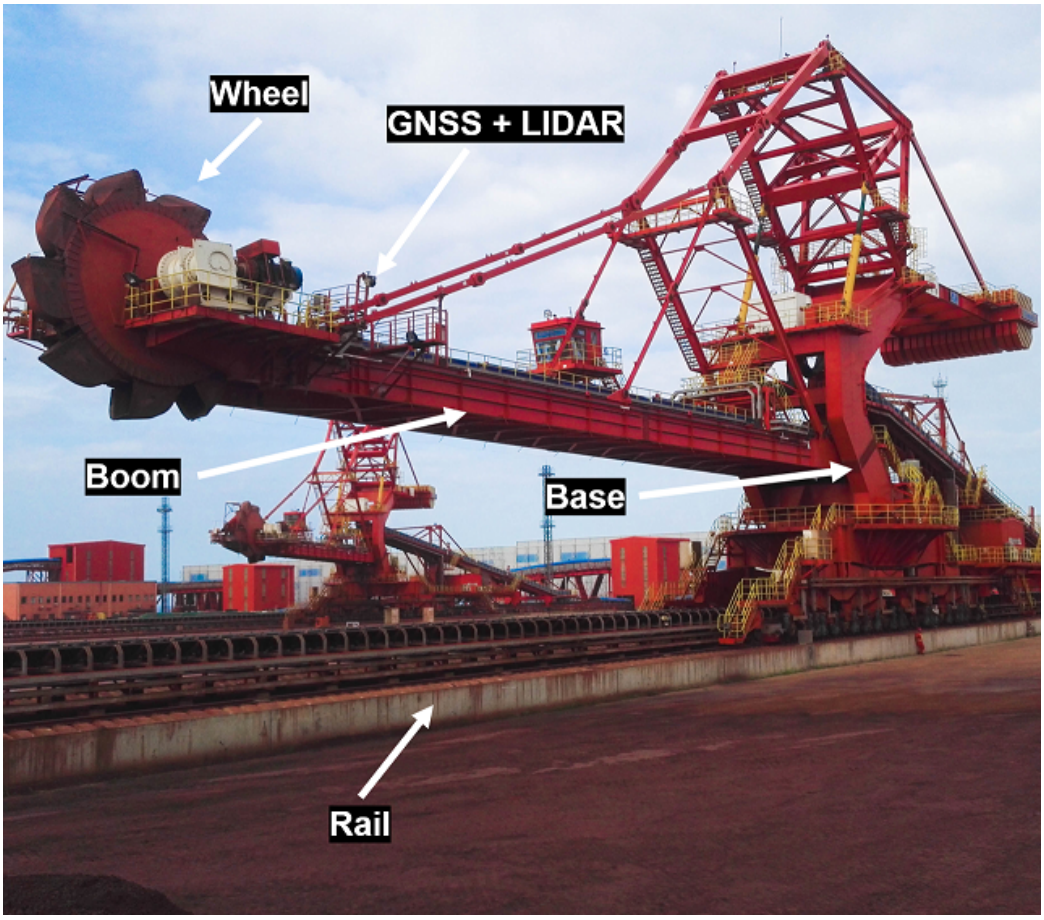


Figure 2.1: BWR platform with sensors.

also contain LIDAR scanners to scan the environment and generate a georectified point cloud model that is used either by an operator or an automatic control system. The georectification process is dependent on extrinsic calibration parameters that quantify the position and attitude offsets between the sensors and the BWR platform. This chapter investigates the problem of calibrating the BWR extrinsic parameters using the GNSS/LIDAR data. After calibration, the georectified point cloud for the material surface will be accurately maintained during operations and the wheel will be capable of more accurate maneuvering

relative to the material surface for material retrieval operations without damage to the stock pile environment.

2.1 System Description

This section discusses the sensors that are mounted and the structure of the Bucket Wheel Reclaimer.

2.1.1 GNSS Receiver

The BWR system has a GNSS receiver with two antennae on each side of the wheel (i.e., 4 antennae and 2 receivers total). Each pair of antennae are mounted approximately parallel to the boom and wheel so that the vector between the antennae allows estimation of the pitch and yaw of the boom [12]. The attitude estimation accuracy of the GNSS receiver is less than 0.9 milli-radian [3]. Each GNSS receiver therefore provides a measurement of the position of each antenna and the pitch and yaw of the platform. While the roll angle of the boom relative to the base is nominally fixed, the roll angle of the platform may vary slightly over time due to mechanical tolerances, twisting of the boom, and other effects. The GNSS antennae on opposite sides of the boom allow measurement of the platform roll as it varies.

2.1.2 LIDAR

The BWR system also has two 2D LIDARs rigidly attached to the boom platform on opposite sides of the wheel. One of the GNSS antennae is rigidly mounted on top of

each LIDAR. See Fig. 2.2. Therefore, the vector from that GNSS antennae to the LIDAR origin is effectively constant.

Each 2D LIDAR has a laser and detector mounted on a rotating disc. These are pulsed LIDAR with range measurement standard deviation, $\sigma_L = 3.8cm$. The BWR wheel and LIDAR axes of rotation are nominally parallel, so that the two LIDAR scan the environment on opposite sides of the BWR wheel. Each LIDAR scans the environment, returning the range and LIDAR rotation angle of reflected points, with measurements defined relative to the origin of the LIDAR frame. As each 2D LIDAR can only detect the environment as it intersects with the LIDAR's plane of rotation, the BWR platform would need to be moved to scan the whole surface. In practical operations, as the boom changes yaw, swinging the wheel through the pile, one LIDAR scans the surface before digging and the other scans the surface after digging.

Because the LIDAR origin moves with the BWR platform, the detected points in each LIDAR scan must be transformed from the LIDAR frame in which they are measured to the fixed Earth frame where they are used to create a 3D point cloud of the scanned environment. This transformation process known as *georectification* is discussed in Section 2.2.4, after defining necessary notation.

The measurements from the two LIDARs are combined to create a single point cloud of the scanned surface. During operations, the surface is time varying due to the BWR's scooping of the material (i.e., reclaiming). Failure to accurately calibrate the extrinsic parameters would result a blurry point cloud of degraded quality.

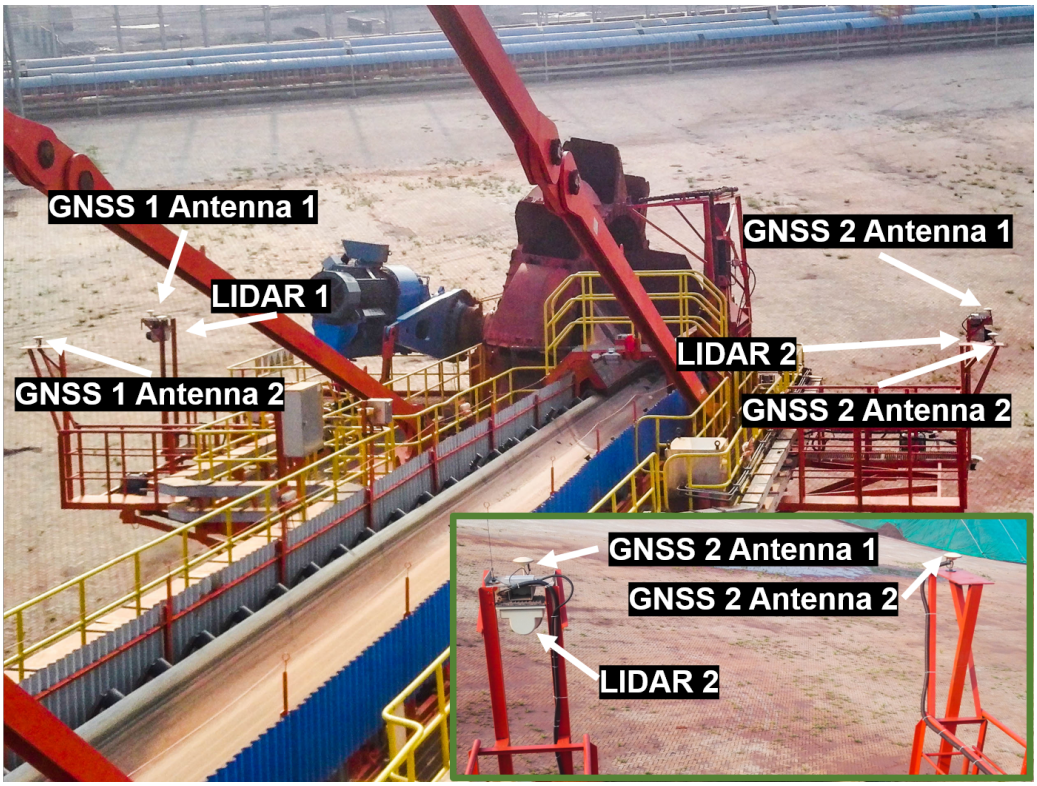


Figure 2.2: The main image is the BWR platform with sensors mounted on it. The inset image magnifies one GNSS/LIDAR system.

2.2 Preliminaries

This section defines reference frames and notation that will be used throughout the dissertation. It also presents the georectification equations.

2.2.1 Frame Definitions

Four reference frames are of interest for this article: earth frame, platform frame, and one frame for each LIDAR. These frames are depicted in Fig. 2.3. The axes of each frame will be defined using the right-handed convention. For a review of reference frames

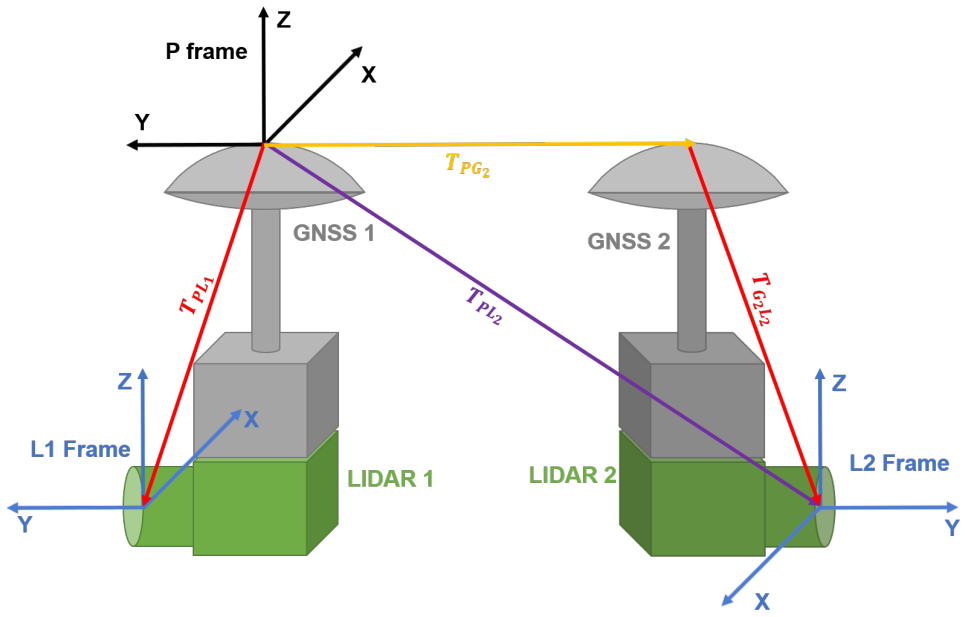


Figure 2.3: Two LIDAR frames (blue) and platform frame (black). The red vectors in P-frame, ${}^{P}T_{PL_1}$ and ${}^{P}T_{G_2L_2}$, are the extrinsic calibration translation vectors estimated in this chapter. The vector ${}^{P}T_{PG_2}$ is measured using GNSS and used to compute ${}^{P}T_{PL_2}$ and the platform roll angle. The vector ${}^{P}T_{PL_2}$ (purple) is computed and used for georectification.

and coordinate systems, see e.g. [16].

- ***E*-frame:** Earth frame is defined using the ENU (East, North, Up) axes, with its origin located at the position of the differential GNSS base station antenna.

- ***P*-frame:** Platform frame origin coincides with GNSS1 Antenna 1, which is mounted immediately above LIDAR 1. The platform frame x-axis points from GNSS 1 Antenna 1 to GNSS 1 antenna 2 (see Fig. 2.2), which are connected to the same receiver on the same side of the wheel. This is the pair of antennae that the GNSS receiver uses to provide a measurement of the platform pitch and yaw. The y-axis is defined by the vector from GNSS 1 Antenna 1 to GNSS 2 Antenna 1 (after removing the component of this vector that is parallel to the platform x-axis). GNSS 2 Antenna 1 is mounted

immediately above LIDAR 2 on the opposite side of the wheel from GNSS 1 Antenna 1. This is the pair of antennae used to compute the platform roll. The z-axis is defined by the right hand rule.

- **L_i-frame (i = 1, 2):** Each LIDAR frame is located at the origin of its LIDAR sensor, which is the effective center of rotation. The y-axis of each LIDAR is defined as the axis of rotation of the LIDAR. The x-axis is defined as the direction at which the LIDAR rotation angle (α) is equal to zero. The z-axis is defined to complete the right-handed system and is (nominally) upward when the platform is level with the ground.

By design the platform $x-z$ plane is approximately parallel to the wheel and to each LIDAR's $x-z$ plane. However, the inability to mechanically align the frames perfectly will leave small rotations that need to be estimated so that computed frame rotations can be accurately performed.

2.2.2 Frame, Point, and Vector Notation

The notation used in this dissertation for vectors and points has special meaning. A vector from point a to point b will be represented as T_{ab}. Any vector has different representations in different reference frames. When it is necessary to represent a vector in a specific frame, a pre-superscript is used. For example, the symbol ${}^{E}T_{ab}$ represents the vector from point a to point b in E-frame.

Transformation of vector representations from one frame to another must account for the relative rotations between the axes of the two frames. For example, ${}^{E}T_{ab} = {}^{E}_{P}R\,{}^{P}T_{ab}$

where ${}^{E}_{P}R$ is the matrix that rotates P-frame such that it aligns with E-frame. The rotation matrix ${}^{E}_{P}R$ is easily computed from the platform roll, pitch, and yaw angles (see Section 2.5.4 in [16]):

$$ {}^{P}_{E}R = \begin{bmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ -s_\psi c_\phi + c_\psi s_\theta s_\phi & c_\psi c_\phi + s_\psi s_\theta s_\phi & c_\theta s_\phi \\ s_\psi s_\phi + c_\psi s_\theta c_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi & c_\theta c_\phi \end{bmatrix} \tag{2.1} $$

where ϕ, θ and ψ are the roll, pitch and yaw sequence of rotations to align P-frame to E-frame and c_ψ and s_ψ are the cosine and sine of the angle ψ. The inverse rotation satisfies ${}^{P}_{E}R = ({}^{E}_{P}R)^\top$.

Similarly points have different coordinates depending on the frame to which they are referenced. The coordinates of a point are the same as the representation of the vector from the frame origin to the point. Therefore, the transformation of the representation of a point between two frames must account for the shift of origin and the rotation between the two frames. A useful example is georectification which is discussed in Section 2.2.4.

2.2.3 Lidar Data

LIDAR L measures the range R and angle α at which its laser emission reflects from an object. This measurement provides the cylindrical coordinates of the reflection point relative to the LIDAR origin:

$$ {}^{L}p = \begin{bmatrix} x \\ 0 \\ z \end{bmatrix} = R \begin{bmatrix} c_\alpha \\ 0 \\ s_\alpha \end{bmatrix}. \tag{2.2} $$

2.2.4 Georectification

Georectification is the process of transforming a LIDAR reflection point ${}^{L}p$ from L-frame to E-frame. The equations for the transformation of points and vectors between L and E-frames, in the form that is most useful for this dissertation, are:

$$ {}^{E}p = {}^{E}T_{EP} + {}^{E}_{P}R\left({}^{P}T_{PL} + {}^{P}_{L}R\ {}^{L}p\right) \tag{2.3} $$

$$ {}^{E}v = {}^{E}_{P}R\ {}^{P}_{L}R\ {}^{L}v. \tag{2.4} $$

Eqns. (2.3-2.4) apply to each LIDAR. When necessary, an i subscript on the L-frame will be added for clarity.

The point ${}^{L}p$ is available from LIDAR measurements as defined in (2.2). The point ${}^{E}p$ represents the same LIDAR reflection point in E-frame. If the reflection point p is fixed in the E-frame, for example a specific point on the wall or ground plane, then each time that point p reflects the LIDAR, ${}^{E}p$ would be constant whereas ${}^{L}p$ would depend on the platform pose (i.e., position and attitude). Lidar measurements occur in L-frame, but are transformed to E-frame to construct a point cloud model of the environment.

Eqn. (2.3) can be viewed as a sequence of two point-transformations. The first point-transformation provides the position of point p in P-frame: ${}^{P}p = {}^{P}T_{PL} + {}^{P}_{L}R\ {}^{L}p$. This transformation uses the rotation ${}^{P}_{L}R$ from L to P-frame and the vector T_{PL} from the origin of P-frame to the origin of L-frame. The quantities ${}^{P}_{L}R$ and T_{PL} are the *extrinsic parameters* of the LIDAR. Because the LIDAR is rigidly attached to the platform the extrinsic parameters are constant in P-frame, but are unknown or inaccurately calibrated. The second point-transformation ${}^{E}p = {}^{E}T_{EP} + {}^{E}_{P}R\ {}^{P}p$ uses the position of the platform origin ${}^{E}T_{EP}$ in E-frame and the rotation ${}^{E}_{P}R$ from P to E, which are both available from

GNSS measurements.

Similarly, eqn. (2.4) breaks the rotation from L to E into two rotations ${}^{E}_{P}R$ which is available from GNSS and ${}^{P}_{L}R$ which will be estimated via calibration. The symbol ${}^{L}v$ represents vector v in L-frame, while ${}^{E}v$ represents the same vector represented in E-frame.

2.2.5 Implementation Detail

When necessary to discuss the translation vector for LIDAR i it will be defined as ${}^{P}T_{PL_i}$. When necessary to discuss the rotation matrix from L_i-frame to P-frame it will be defined as ${}^{P}_{L_i}R$. Throughout the chapter, when talking about issues that apply to both LIDAR's, the subscript i will be dropped.

LIDAR 2 is located on the opposite side of the bucket wheel from GNSS 1 Antenna 1, which defines the origin of P-frame. The translation vector ${}^{P}T_{PL_2}$ is approximately 13 m in length. When the BWR is in use, machine torques that spin the wheel may cause bending that would negate our assumption that ${}^{P}T_{PL_2}$ is constant. Therefore, ${}^{P}T_{PL}$ is decomposed as

$$ {}^{P}T_{PL_2} = {}^{P}T_{PG_2} + {}^{P}T_{G_2L_2} $$

where ${}^{P}T_{G_2L_2}$ is short and (assumed to be) constant and ${}^{P}T_{PG_2}$ is measured by the GNSS sensors. As any flexing of the platform is reflected in the measurement of ${}^{P}T_{PG_2}$, this modeling of the translation vector ${}^{P}T_{PL_2}$ conforms better with our assumption that the extrinsic parameters that we will estimate (i.e., ${}^{P}_{L_1}R$, ${}^{P}_{L_2}R$, ${}^{P}T_{G_2L_2}$, and ${}^{P}T_{G_1L_1}$) are constant.

2.2.6 Extrinsic Parameter Notation

The error models for the calibration parameters are

$$ {}^{P}\hat{T}_{PL} = {}^{P}T_{PL} + \delta T \tag{2.5} $$

$$ {}^{P}_{L}\hat{R} = {}^{P}_{L}R(I - [\rho\times]) \tag{2.6} $$

where ${}^{P}T_{PL}$ and ${}^{P}_{L}R$ are the true translation and rotation, ${}^{P}\hat{T}_{PL}$ and ${}^{P}_{L}\hat{R}$ are the estimated translation and rotation, and δT and ρ parameterize the error between ${}^{P}\hat{T}_{PL}$ and ${}^{P}_{L}\hat{R}$ and their true values. The multiplicative form of eqn. (2.6) is derived in, e.g. Section 10.5 of [16]. Given initial (e.g., manually calibrated) values for ${}^{P}\hat{T}_{PL}$ and ${}^{P}_{L}\hat{R}$, this chapter uses LIDAR and GNSS measurement data from the natural BWR environment to compute estimates $\delta\hat{T}$ and $\hat{\rho}$. Once $\delta\hat{T}$ and $\hat{\rho}$ are available, based on eqns. (2.5-2.6), the following equations are used to correct the initial values:

$$ {}^{P}\hat{T}^{+}_{PL} = {}^{P}\hat{T}_{PL} - \delta\hat{T} \tag{2.7} $$

$$ {}^{P}_{L}\hat{R}^{+} = {}^{P}_{L}\hat{R}(I + [\hat{\rho}\times]) \tag{2.8} $$

where the right superscript '+' denotes the corrected value. This process is iterated until convergence with the corrected value of the last iteration serving as the starting point for the next iteration.

As a convenience, the dissertation will use the following shorthand notation: $\mathbf{x} = [T_{PL}, {}^{P}_{L}R]$ and $\delta\mathbf{x} = [\delta T,\ \rho]$.

Chapter 3

Calibration

This chapter discusses the extrinsic calibration technique for the Bucket Wheel Reclaimer.

3.1 GPS to LIDAR Extrinsic Calibration

This chapter presents a method to calibrate the extrinsic parameters of LIDARs mounted on a Bucket Wheel Reclaimer (BWR). BWR's are widely used for stacking and reclaiming bulk materials in stockyards. Current BWR systems are either manually operated, remotely operated or semi-automated using the real-time point cloud data generated by one or more LIDARs. Accurate calibration is of crucial importance as the accuracy of the Earth frame point cloud data depends on it. Automated calibration is also a pre-requisite for fully autonomous BWR control. Calibration of BWR systems are more difficult than many other LIDAR systems because of their limited pose variation capabilities and the environmental constraints of stockyards. This chapter analyzes the problem, presents observability condi-

tions, presents a method to estimate the calibration parameters and discusses the challenges involved with BWR systems. The results demonstrate subdecimeter accuracy.

3.1.1 Literature Review

Lu presents the ground work for treating a BWR as a robotic arm for operation automation[29]. This paper works with an encoder based BWR and does not address the extrinsic parameter calibration problem. Zhang et al. [47] use data from a GNSS/LIDAR system that detects markers pre-located at surveyed locations in the environment to calibrate the BWR boom length and yaw errors relative to their manually calibrated values. Article [47] does not calibrate the extrinsic parameters as defined herein. The approach presented herein uses the natural BWR environment without presurveyed markers.

Fernandez et al. [17] and Choi et al. [11] present methods to perform relative calibration between multiple 2D LIDARs without using GNSS. These methods require LIDAR systems with overlapping scan areas and significant pose variation of the LIDAR system. BWR systems do not have overlapping fields of view and are not capable of the necessary pose variation. Underwood et al. [42] and He et al. [23] present methods to perform calibration of multiple 2D LIDARs mounted on a vehicle using structures naturally present in the scene, but also require the sensors to view the same environment or at-least overlapping sections. Gao et al. [21] proposes a calibration method between GNSS and a 2D LIDAR using reflective tape as targets at unknown locations that are fixed in Earth frame (i.e., unsurveyed landmarks). Detections from different LIDAR locations and poses allow simultaneous estimation of landmark locations and extrinsic parameters. Implementation of this approach requires LIDAR remission measurements to distinguish the reflective tape

from other surfaces in the environment.

Several articles present methods to estimate extrinsic parameters between a 2D-LIDAR and a rotating unit center [4]. Zeng et al. [45] present a method to estimate the bias angle of a 2D LIDAR and a Pan Tilt Unit (PTU) using a horizontal plane placed on top of the unit. Kang et al. [24] present a method to estimate the extrinsic calibration parameters between a LIDAR and a PTU by placing a high number of non-horizontal target planes on the environment. Yamao et al. [44] estimates the roll and tilt angle between a 2D LIDAR and PTU without any plane or other environmental requirements. The method minimizes the error between repeated detections of points, with the assumption that the LIDAR observes the same 3D point from two different angles during a single rotation. It does not estimate all 6 extrinsic calibration parameters. None of these approaches apply to the BWR problem defined herein, due to limited pose variation of the BWR (slowly moving base, fixed roll, limited pitch) and zero overlap between the 2D LIDARs on every scan.

There is also work on LIDAR system calibration for applications other than BWR [28, 31, 17, 11, 42, 23, 21]. Levinson et al. [28] solves the extrinsic parameter calibration between a GNSS and a 3D LIDAR by minimizing an energy function computed from consecutive pairs of scan lines. Muhammad et al. [31] proposed a method to calibrate the intrinsic parameters (parameters that define the position and orientation of each of the laser beams) of a 3D LIDAR using general planar structures with unknown parameters. The method requires dedicated calibration targets and various manual measurements. The above papers solve the calibration of a 64 laser 3D LIDAR which, from a static position, can scan a 3D volume with much higher scan density than is possible with a planar 2D LIDAR.

In contrast, the approach presented herein works with one or more 2D LIDARs and does not require special markers (e.g., reflective tape) or surveyed locations (other than a DGNSS base station antenna). The only assumption is that the BWR environment contains an approximately horizontal (e.g., ground) plane and an approximately vertical (e.g., rail wall) plane[1]. The parameters of the planes are not known, but are estimated as part of the process.

3.1.2 Problem Statement

The objective of the chapter is the estimation of the extrinsic parameters ${}^{P}T_{PL}$ and ${}^{P}_{L}R$ for each LIDAR.

A method is proposed that computes a cost using geometric features extracted from measurements of the environment and then optimizes the extrinsic parameters by numerically minimizing that cost. The geometric features that are used in this chapter are the E-frame normals and distances of the planes that are naturally present in the stockyard environment.

Due to the fact that the BWR base moves along a rail that is set above the nearly horizontal (but slightly sloped) plane that forms the stockyard, many scans contain (at least) two lines of reflections that are from surfaces fixed in E-frame. These two surfaces are the horizontal stock ground yard plane and the vertical plane defined by the wall that forms the platform on which the BWR rail is mounted. Denote these planes as

$$
\begin{aligned}
{}^{E}\pi_{H} &= \left\{ {}^{E}p \mid {}^{E}N_{H} \cdot {}^{E}p = {}^{E}d_{H} \right\} \text{ and} \\
{}^{E}\pi_{V} &= \left\{ {}^{E}p \mid {}^{E}N_{V} \cdot {}^{E}p = {}^{E}d_{V} \right\}
\end{aligned}
$$

[1]The approach easily generalized to any other pair of planes with significantly different normals.

where ${}^{E}N_H$ ($\approx [0,\ 0,\ 1]^\top$) and ${}^{E}N_V$ ($\approx [0,\ 1,\ 0]^\top$) are the unit normals to the ground and wall planes and ${}^{E}d_H$ and ${}^{E}d_V$ are the distances of these planes from the origin of E-frame. These horizontal and wall planes are used in the calibration; therefore, reliable and efficient extraction of these plane points from each scan is important.

If $\mathbf{y}$ denotes the LIDAR and GNSS measurements and $\{\mathbf{T}, \mathbf{R}\}$ represents the extrinsic parameters for a GNSS-LIDAR pair, then the optimal parameters minimizes the cost

$$\mathbf{T}^*, \mathbf{R}^* = \underset{\mathbf{T},\mathbf{R}}{\operatorname{argmin}}\, C\left(\mathbf{T}, \mathbf{R} : \mathbf{y}\right). \tag{3.1}$$

This chapter will formulate the cost function (see eqn. (3.12)), analyze the requirements for a unique minimum to exist and discuss how to solve for $\mathbf{T}^*$ and $\mathbf{R}^*$. The problem is divided into four steps:

A. Extract two sets of LIDAR return data S_H and S_V:

$$\begin{aligned} S_H &= \left\{ {}^{L}p_i \middle| {}^{L}p_i \text{ is on } {}^{E}\pi_H \right\} \text{ and} \\ S_V &= \left\{ {}^{L}p_i \middle| {}^{L}p_i \text{ is on } {}^{E}\pi_V \right\}. \end{aligned}$$

The two sets S_H and S_V will be constructed to be non-intersecting.

B. Segment $S_H = \cup S_{H_i}$ and $S_V = \cup S_{V_i}$ such that

$$\begin{aligned} S_{H_i} &= \left\{ {}^{L}p_i, {}^{L}p_j \in S_H \middle| \max_{i,j} \| {}^{E}p_i - {}^{E}p_j \| < L \right\} \text{ and} \\ S_{V_i} &= \left\{ {}^{L}p_i, {}^{L}p_j \in S_V \middle| \max_{i,j} \| {}^{E}p_i - {}^{E}p_j \| < L \right\}. \end{aligned}$$

Note that $S_{H_{i_1}}$ and $S_{H_{i_2}}$ will be constructed to be non-intersecting, as are $S_{V_{i_1}}$ and $S_{V_{i_2}}$, for $i_1 \neq i_2$.

C. Split each $S_{H_i} = S^f_{H_i} \cup S^r_{H_i}$ where $S^f_{H_i}$ will be used to estimate $\hat{N}_{H_i}$ and $\hat{d}_{H_i}$. Each S_{V_i} will be split similarly.

D. Given the estimated normals and distances $\hat{N}_{H_i}$ and $\hat{d}_{H_i}$, use $S^r_{H_i}$ and $S^r_{V_i}$ to estimate the extrinsic calibration parameters.

After scanning the environment, Steps A and B are performed once. Then for each planar patch S_{H_i} or S_{V_i}, using the current extrinsic parameters, Step C is performed to estimate each patch's normal and distance. Then using these estimated plane parameters, the extrinsic calibration parameters are estimated. Therefore, Steps A and B are data processing necessary so that a proper cost function can be defined and minimized in Steps C and D. Then, starting from nominal extrinsic calibration parameters, Steps C and D are iterated until convergence.

3.1.3 Solution Approach

This section discusses and analyzes each step of the solution approach.

3.1.3.1 Plane Point Extraction

When a LIDAR scan intersects a portion of a plane that is stationary in E-frame, the reflections define a sequence of points along a line in E-frame. The motion of the platform is sufficiently slow and the period of a scan is sufficiently short that this line in E-frame also appears as a line in L-frame. Therefore, a line segment extraction algorithm is used to extract the plane points in L-frame.

As the BWR base moves along a rail that is set above π_H and π_V, many scans con-

tain at least two lines of reflections that are from those planes, along with other reflections. For line segment extraction, the field of view of the LIDAR is chosen such that reflections from the BWR base, rail and horizontal plane are captured, while most of the overhead reflections from the super-structure are discarded. This choice of the field of view enables extraction of line segments from π_H and π_V sequentially, while discarding unwanted points reflected from other surfaces. For each LIDAR and each scan, the goal is to extract two sequences of points ${}^L p_j$ that form line segments ${}^L V_{H_l}$ and ${}^L V_{V_l}$ in L-frame that are reflect from π_H and π_V respectively. Denote these sets of line segments as ${}^L\mathbf{V}_H$ and ${}^L\mathbf{V}_V$ where

$$\begin{aligned}
{}^L\mathbf{V}_H &= \left\{ {}^L V_{H_l} \;\middle|\; l = 1, 2, \cdots, n_L \right\} \text{ and} \\
{}^L\mathbf{V}_V &= \left\{ {}^L V_{V_l} \;\middle|\; l = 1, 2, \cdots, n_L \right\}, \text{ where} \\
{}^L V_{H_l} &= \left\{ {}^L p_j \;\middle|\; j = 1, 2, \cdots, n_{J_l} \right\} \text{ and} \\
{}^L V_{V_l} &= \left\{ {}^L p_k \;\middle|\; k = 1, 2, \cdots, n_{K_l} \right\}.
\end{aligned}$$

In these definitions, the symbol n_L represents the total number of the line segments extracted for each plane. The symbols n_{J_l} and n_{K_l} represent the number of points in the l-th line segments ${}^L V_{H_l}$ and ${}^L V_{V_l}$ respectively.

Various line extraction algorithms exist: RANSAC [18], Hough transform [19], incremental line extraction [19], etc. The tradeoffs between such algorithms are discussed in e.g. [32]. The incremental line extraction algorithm has been used in many applications[43, 40, 37]. It is simple and efficient as well. This is especially true in this application where the two lines desired from each scan are known to intersect and nearly be orthogonal, because the planes ${}^E\pi_H$ and ${}^E\pi_V$ intersect and are approximately orthogonal. The fact that the lines intersect facilitates their sequential extraction with the incremental algorithm. The

Algorithm 1 Incremental Line Extraction

Objective: Given a set of points ${}^{L}P_l$ from the LIDAR's l-th scan, extract line segments ${}^{L}V_{H_l}$ and ${}^{L}V_{V_l}$ that are from π_H and π_V respectively.

1. Add the first two points to the first line segment ${}^{L}V_{H_l}$. Extract line parameters for ${}^{L}V_{H_l}$.

2. Compute residual r for the next point in ${}^{L}P_l$ with the current line parameters.

3. If the $|r| < \epsilon$, add the point to ${}^{L}V_{H_l}$, extract and update the line parameters and go to Step 2.

4. Otherwise do not include the point, return the line points.

5. Continue with the next 2 points for the second line segment ${}^{L}V_{V_l}$ extraction following Steps $1 - 4$.

6. If ${}^{L}V_{H_l}$ is approximately orthogonal to ${}^{L}V_{V_l}$, keep the extracted line segments, otherwise discard them as outliers. Repeat for the next scan.

fact that the two line segments are expected to be orthogonal facilitates detection and rejection of outliers. The algorithm[2] is summarized in Algorithm 1. Line extraction is performed and the extracted data is saved in L-frame; therefore, the scans are not affected by extrinsic parameter errors.

Two example line extraction results are shown in Fig. 3.1. The figure shows

[2]The parameter $\epsilon = 2\sigma_L$ where $\sigma_L = 3.8cm$ is the manufacuter's stated range measurement standard deviation.

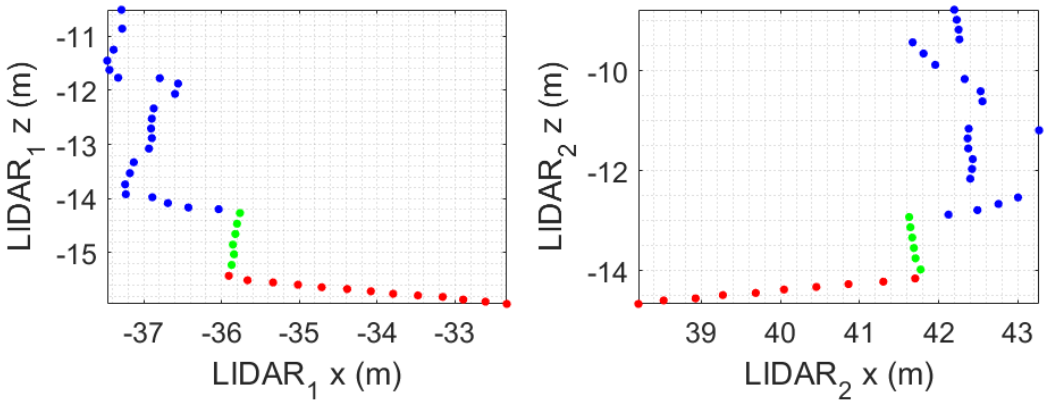


Figure 3.1: Line extraction from LIDAR scan data. Red points indicate the first extracted line which belongs to the ground plane. The green points indicate the second line which belongs to the vertical plane. The blue points from the BWR movable platform are discarded.

scans from two LIDARs and points belonging to lines that are extracted from the scans. Algorithm 1 extracts the red and the green points corresponding to π_H and π_V, respectively. The blue points, which are reflected by the BWR platform, are discarded.

After all the lines are extracted, the set of all points on any horizontal or vertical line segment, S_H and S_V are formed:

$$S_H = \bigcup_{l=1}^{n_L} {}^L V_{H_l} \quad \text{and} \quad S_V = \bigcup_{l=1}^{n_L} {}^L V_{V_l}. \tag{3.2}$$

The union is over all line segments. Note that the points in S_H amd S_V are saved in LIDAR frame, so unaffected by the extrinsic parameters.

3.1.3.2 Plane Decomposition into Patches

The actual ground and wall are not truly planar on a larger scale, which necessitates their subdivision into smaller patches. The objective is to divide S_H and S_V into

local patches S_{H_i} and S_{V_i} such that

$$S_H = \bigcup_{i=1}^{n_H} S_{H_i} \quad \text{and} \quad S_V = \bigcup_{i=1}^{n_V} S_{V_i}.$$

where each local patch S_{H_i} and S_{V_i} is small enough to be accurately represented by a plane. The requirements for S_{H_i} and S_{V_i} are defined in Step B in Section 3.1.2. This decomposition is straightforward to implement. Our approach specifies a center point ${}^{E}p_{c_i}$ for each patch in S_H and S_V. Points in S_H within a radius L of ${}^{E}p_{c_i}$ are sorted into S_{H_i}, S_{V_i}'s are sorted similarly.

3.1.3.3 Data Splitting for Plane Parameter Estimation

If the extrinsic calibration parameters were correct, then the E-frame plane-fit residual for each patch would have a zero mean Gaussian distribution with small standard deviation determined primarily by the LIDAR range measurement noise. If the extrinsic parameters are incorrect the residual distribution may be biased, have higher standard deviation, or be non-Gaussian (see e.g. Fig. 3.5). In this step, the data for each patch is split into two sets $S_{H_i} = S^f_{H_i} \cup S^r_{H_i}$. The set $S^f_{H_i}$ is used to estimate plane parameters. The other set $S^r_{H_i}$ is used to compute a cost function for estimation of the calibration parameters. Let P represent a planar patch (i.e. one set in either S_H or S_V).

There are many ways to partition the set of points P into the two subsets P^f and P^r. The manner in which the patch is split affects the numeric properties of the resulting extrinsic parameter estimation problem. The goal of this splitting to enhance the sensitivity of the cost to the extrinsic parameter error. The effect of extrinsic parameter error and pose variation on the georectified points and the estimated normal is exemplified

in Fig. 3.2. The top-left figure shows georectified plane points using the optimized extrinsic calibration parameters and the corresponding estimated normal. The scans were performed while varying platform pose. The top-right figure shows the same L-frame points after transformation to E-frame (i.e., georectified) when there is error in the extrinsic calibration parameters. These calibration errors cause the E-frame planar patch to be non-planar and the estimated normal for the patch to deviate from the true normal for the plane patch. To estimate the extrinsic parameter vector, our goal will be to define a cost function that is sensitive to the error in that vector, that is also suitable for numeric optimization, and that yields a set of consistent planar patches for the optimal extrinsic parameters. The bottom row of Fig. 3.2 illustrates that different partitioning approaches yield different normals in Step C, which yield cost functions in Step D with different sensitivities to the extrinsic parameter error vector. In Fig. 3.2 points from $S^f_{H_i}$ are red and $S^r_{H_i}$ are blue. This sensitivity is analyzed next.

Suppose, a patch $P \in S_H \cup S_V$ is split into two non-intersecting point sets P^f and P^r. Let $\hat{N}_f$ and $\hat{d}_f$ be the plane parameters computed from P^f (in E-frame). The residual for a point $p \in P^r$ (in E-frame) relative to the plane defined by $\hat{N}_f$ and $\hat{d}_f$ is

$$r_f(p) = \hat{N}_f \cdot p - \hat{d}_f. \tag{3.3}$$

Consider the cost function for patch P:

$$C_P(\mathbf{x} : P^f, P^r) = \sum_{p_i \in P^r} \left(r_f(p_i)\right)^2$$

which should be read as the cost of $\mathbf{x}$ computed over points in P^r relative to the plane that is fit to points in P^f. In this expression $\mathbf{x}$ represents the value of the calibration parameters

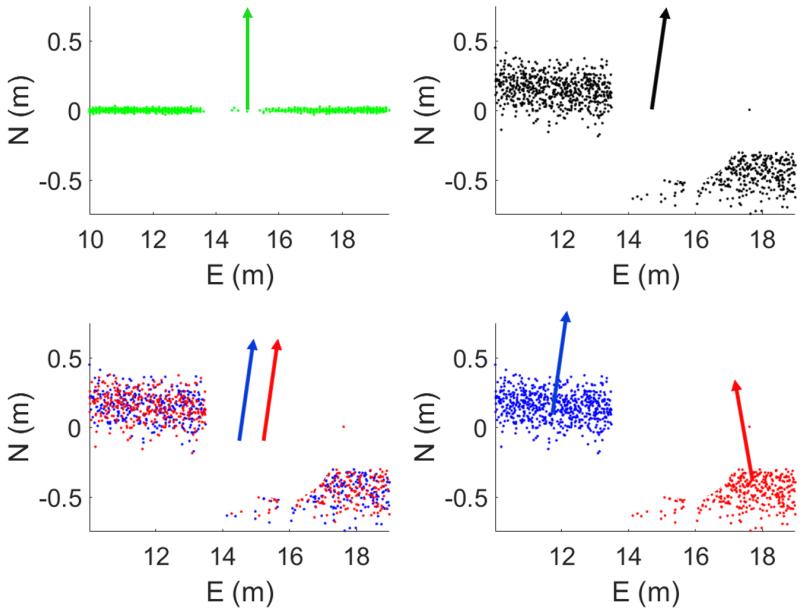


Figure 3.2: Georectified points and estimated plane normals. (Top-left) Georectified points with correct (i.e. final) extrinsic parameter. (Top-right) Georectified points with nominal extrinsic parameter. (Bottom-left) Georectified points with nominal extrinsic parameter and odd-even splitting. (Bottom-right) Georectified points with nominal extrinsic parameter with pose-based splitting.

is defined in Section 2.2.6. When the context is clear, this will be shortened to $C_P(\mathbf{x})$.

This cost function can be manipulated as follows:

$$\begin{aligned} C_P(\mathbf{x}) &= \sum_{p_i \in P^r} \left(\hat{N}_f^\top p_i - \hat{d}_f \right) \left(p_i^\top \hat{N}_f - \hat{d}_f \right) \\ &= \sum_{p_i \in P^r} \left(\hat{N}_f^\top p_i - \hat{N}_f^\top \bar{p}_f \right) \left(p_i^\top \hat{N}_f - \bar{p}_f^\top \hat{N}_f \right) \\ &= \sum_{p_i \in P^r} \hat{N}_f^\top \left(p_i - \bar{p}_f \right) \left(p_i - \bar{p}_f \right)^\top \hat{N}_f \\ &= \hat{N}_f^\top \left(\sum_{p_i \in P^r} \left(p_i - \bar{p}_f \right) \left(p_i - \bar{p}_f \right)^\top \right) \hat{N}_f \end{aligned} \tag{3.4}$$

where $\bar{p}_f$ is the centroid of P^f and we have used the fact that for a plane fit to P^f it will be the case that $\hat{d}_f = \hat{N}_f^\top \bar{p}_f$. Note that in $C(\mathbf{x})$ each of p_i, $\bar{p}_f$, $\hat{N}_f$, and $\hat{d}_f$ are in E-frame, so affected by $\mathbf{x}$.

Standard methods for estimation of plane parameters from a set S of points on the plane can be divided into two different categories: averaging and optimization. Optimization based methods such as Principle Component Analysis (PCA) and Singular Value Decomposition (SVD) outperform averaging methods both in computational performance and quality [26]. For $p_i \in S$ the mean is

$$\bar{p}_S = \frac{1}{n} \sum_{i=1}^{n} p_i.$$

and the covariance matrix is

$$\Sigma_S = \frac{1}{n-1} \left(D - \bar{p}_S \right)^\top \left(D - \bar{p}_S \right)$$

where $D = [p_1, p_2, \cdots p_n]^\top$ and n represents the cardinality of S. In the PCA approach [26, 5], Σ_S is decomposed into its eigenvectors and eigenvalues. The eigenvectors with the

largest two eigenvalues are the directions of maximum variance of the data. The vector orthogonal to these vectors, which is also the eigenvector corresponding to the smallest eigenvalue, is the estimated normal to the plane $\hat{N}_S$. The distance parameter is $\hat{d}_S = \hat{N}_S^\top \bar{p}_S$.

If the distribution of points in the two sets P^f and P^r are similar in the sense that their centroids and covariance matrices are nearly equal (i.e., $\bar{p}_f \approx \bar{p}_r$ and $\Sigma_f \approx \Sigma_r$), then eqn. (3.4) becomes

$$\begin{aligned} C_P(\mathbf{x}) &\approx \hat{N}_f^\top \sum_{p_i \in P^r} (p_i - \bar{p}_r)(p_i - \bar{p}_r)^\top \hat{N}_f \\ &\approx \hat{N}_f^\top \Sigma_r \hat{N}_f \\ &\approx 0. \end{aligned} \tag{3.5}$$

This is true for all $\mathbf{x}$. This shows that an unfortunate choice of the two sets P^f and P^r will lead to a cost function that is not sensitive to the error in the extrinsic parameter error. The bottom left figure of Fig. 3.2 shows such an example of such a case, in which the blue points (i.e., P^r) are from even scans and the red points (i.e., P^f) are from odd scans. This splitting approach yields very similar points in each set because the platform pose changes little from one scan to the next. Very similar point distributions yield approximately the same plane parameters from each set.

To avoid this unfortunate circumstance, each patch should be decomposed in such a way that $\hat{N}_f \neq \hat{N}_r$ and $\hat{d}_f \neq \hat{d}_r$, unless the calibration error is zero. The bottom right figure of Fig. 3.2 shows data splitting based on platform pose, which yields significantly different normals and as a result, a more sensitive cost function. The following paragraphs discuss the effect of platform pose on the georectified points and estimated normals, which gives insight on data splitting criteria for a patch.

Given a point ${}^{L}p$ in L-frame, the computed georectified point in E-frame using the nominal extrinsic calibration parameter is

$$\begin{aligned} {}^{E}p &= {}^{E}T_{EP} + {}^{E}_{P}R\left({}^{P}T_{PL} + {}^{P}_{L}R\ {}^{L}p\right) \\ &= {}^{E}T_{EP} + {}^{E}_{P}R\left({}^{P}\hat{T}_{PL} - \delta T + {}^{P}_{L}\hat{R}(I + [\rho\times]){}^{L}p\right) \end{aligned} \tag{3.6}$$

where $[\rho\times]$ is the skew-symmetric representation of vector ρ and has the form:

$$[\rho\times] = \begin{bmatrix} 0 & -\rho_3 & \rho_2 \\ \rho_3 & 0 & -\rho_1 \\ -\rho_2 & \rho_1 & 0 \end{bmatrix}.$$

Eqn. (3.6) yields,

$$\begin{aligned} {}^{E}p &= {}^{E}\hat{p} - {}^{E}_{P}R\left(\delta T - {}^{P}_{L}\hat{R}[\rho\times]\ {}^{L}p\right) \\ &= {}^{E}\hat{p} + {}^{E}\tilde{p}, \end{aligned} \tag{3.7}$$

where

$$\begin{aligned} {}^{E}\hat{p} &\doteq {}^{E}T_{EP} + {}^{E}_{P}R\left({}^{P}\hat{T}_{PL} + {}^{P}_{L}\hat{R}\ {}^{L}p\right) \text{ and} \\ {}^{E}\tilde{p} &= -{}^{E}_{P}R\left(\delta T - {}^{P}_{L}\hat{R}[\rho\times]\ {}^{L}p\right) \\ &= -{}^{E}_{P}R\left(\delta T + {}^{P}_{L}\hat{R}[{}^{L}p\times]\ \rho\right) \qquad (3.8) \\ &= -\left[{}^{E}_{P}R,\ \ {}^{E}_{P}R\ {}^{P}_{L}\hat{R}\ [{}^{L}p\times]\right]\delta\mathbf{x}. \qquad (3.9) \end{aligned}$$

The symbol ${}^{E}\hat{p}$ denotes the computed E-frame coordinates, ${}^{E}p$ denotes the true E-frame coordinate, and ${}^{E}\tilde{p}$ denotes the error due to the extrinsic parameter error $\delta\mathbf{x}$ as defined in Section 2.2.6.

For any given planar patch P, assuming that the diameter $2L$ of the patch is small relative to its distance to the LIDAR, all points ${}^{L}p_i \in P$ will have very similar coordinates. This variation in ${}^{L}p_i$ helps (mildly) with the observability of ρ, but based on eqn. (3.9), does not help at all with the observability of δT. Note that ${}^{P}_{L}R$ is constant. To achieve significantly different effects of the parameter error for a given patch (i.e., to achieve observability), the patch must be scanned from distinct poses, which changes ${}^{E}_{P}R$. Therefore, $\hat{P}^f$ should be defined using points from one pose, while $\hat{P}^r$ is defined using points from a distinctly different pose.

Note that the error in the estimate of a normal is due to the portion of the error in each georectified point ${}^{E}\hat{p}_i$ that is in the direction of the normal. The portion of the error that is perpendicular to the normal (i.e., in the plane) has no effect. Therefore, the platform pitch (see eqns. (2.1) and (3.9)) strongly affects the estimated normal of the ground plane, while the effect of platform yaw is weak. Alternatively, the platform yaw strongly affects the estimated normal of the vertical plane, while the effect of platform pitch is weak.

3.1.3.4 Extrinsic calibration parameters estimation

The overall cost function is defined by summing over all patches of all planes:

$$C(\mathbf{x}) = \sum_{P \in S_H \cup S_V} C_P(\mathbf{x} : P^f, P^r). \tag{3.10}$$

Given the estimated plane parameters for the various horizontal and vertical planes and the calibration parameters $\hat{\mathbf{x}} = [\hat{T}_{PL}, {}^{P}_{L}\hat{R}]$, the residuals for P are computed as

$$\mathbf{r}_P(\hat{\mathbf{x}}) = \left\{ r_k \mid r_k = \hat{N}_f \cdot {}^{E}p_k - \hat{d}_f \text{ for all } {}^{E}p_k \in P \right\} \tag{3.11}$$

for each $P \in S_H \cup S_V$. Then the cost of eqns. (3.1) or (3.10) is

$$C(\mathbf{x}) = \sum_{P \in S_H \cup S_V} \|\mathbf{r}_P(\mathbf{x})\|^2 \tag{3.12}$$

where $\mathbf{r}_P$ defined in eqn. (3.11) is treated as a vector. The cost function $C(\mathbf{x})$ is nonlinear in the extrinsic parameter vector $\mathbf{x}$. The extrinsic parameter vector is estimated by minimizing the cost defined in eqn. (3.12) iteratively. Each iteration solves a linearized problem yielding the extrinsic parameter error vector $\delta\mathbf{x} = [\delta T,\ \rho]$, which are accumulated using eqns. (2.7-2.8). Iterations terminate when $\delta\mathbf{x}$ is sufficiently small. The residual vector is analyzed in the next paragraph.

Let the environment contain a plane with normal ${}^E\hat{N}$ and distance ${}^E\hat{d}$. Residual for any point ${}^Ep$ relative to the plane is

$$r_P = {}^E\hat{N} \cdot {}^Ep - {}^E\hat{d}.$$

Replacing ${}^Ep$ using (3.7), the plane residual becomes

$$r_P = {}^E\hat{N} \cdot \left({}^E\hat{p} + {}^E\tilde{p}\right) - {}^E\hat{d}$$

Using (3.8),

$$\begin{aligned} r_P &= {}^E\hat{N} \cdot {}^E\hat{p} - {}^E\hat{d} + {}^E\hat{N} \cdot {}^E\tilde{p} \\ &= {}^E\hat{N} \cdot {}^E\hat{p} - {}^E\hat{d} - {}^E\hat{N}^\top {}^E_P R \left(\delta T + {}^P_L\hat{R}[{}^Lp\times]\ \rho\right) \\ &= \hat{r}_P - {}^E\hat{N}^\top {}^E_P R \left(\delta T + {}^P_L\hat{R}[{}^Lp\times]\ \rho\right). \end{aligned} \tag{3.13}$$

The first term of eqn. (3.13) is the residual for point ${}^E\hat{p}$ relative to the plane with the estimated normal ${}^E\hat{N}$ and distance ${}^E\hat{d}$. For multiple measurements, the above equation can

be written in matrix form as:

$$\mathbf{r}_P(\mathbf{x}) = \mathbf{r}_P(\hat{\mathbf{x}}) + \mathbf{H}\delta\mathbf{x} \tag{3.14}$$

where $\mathbf{H}$ is the Jacobian matrix of $\mathbf{r}_P(\mathbf{x})$ evaluated at $\hat{\mathbf{x}}$. The Jacobian matrix has the form

$$\mathbf{H} = -\left[\begin{array}{cc} {}^E\hat{N}_H^\top \, {}^E_P R & {}^E\hat{N}_H^\top \, {}^E_P R \, {}^P_L\hat{R} \, [{}^L p_\times] \\ \vdots & \vdots \\ \hline {}^E\hat{N}_V^\top \, {}^E_P R & {}^E\hat{N}_V^\top \, {}^E_P R \, {}^P_L\hat{R} \, [{}^L p_\times] \\ \vdots & \vdots \end{array}\right]. \tag{3.15}$$

The top set of rows correspond to the points on patches of the horizontal plane, while the bottom set of rows correspond to points on patches of the vertical plane.

The BWR platform cannot change its roll angle. But the roll varies slightly from zero due to mechanical tolerances and twisting of the boom. For simplicity in the following, we will assume the roll angle is zero. Similar analysis and conclusions would result for any other constant roll angle. Any variation in the roll angle increases the diversity of the columns and improves the observability of the problem (i.e., rank of the Jacobian). The BWR can vary platform yaw in the range $\psi \in [70°,\ 170°]$ and pitch in the range $\theta \in [-10°, 10°]$. Therefore, small angle approximation is valid for pitch. With these assumptions, the rotation matrix ${}^E_P R$ defined in eqn. (2.1) simplifies to

$${}^E_P R = \left[\begin{array}{ccc} c_\psi & -s_\psi & \theta c_\psi \\ s_\psi & c_\psi & \theta s_\psi \\ -\theta & 0 & 1 \end{array}\right]. \tag{3.16}$$

The following analysis is for LIDAR 1. The analysis for LIDAR 2 would be almost identically, so its is not included.

A row of $\mathbf{H}$ corresponding to a patch of the horizontal plane π_H with normal ${}^{E}N_H \approx [0,\ 0,\ 1]$ has the form

$$\begin{aligned}\mathbf{H}_H &= -\left[{}^{E}\hat{N}_H^{\top}\ {}^{E}_{P}R,\quad {}^{E}\hat{N}_H^{\top}\ {}^{E}_{P}R\ {}^{P}_{L}\hat{R}\ [{}^{L}p_{\times}]\right] \\ &= -\left[\begin{bmatrix} -\theta, & 0, & 1 \end{bmatrix}, \begin{bmatrix} -\theta, & 0, & 1 \end{bmatrix} {}^{P}_{L}\hat{R}\ [{}^{L}p_{\times}]\right] \end{aligned} \tag{3.17}$$

A row of $\mathbf{H}$ corresponding to a patch of the vertical plane π_V with normal ${}^{E}N_V \approx [0,\ 1,\ 0]$ has the form

$$\begin{aligned}\mathbf{H}_V &= -\left[{}^{E}\hat{N}_V^{\top}\ {}^{E}_{P}R,\quad {}^{E}\hat{N}_V^{\top}\ {}^{E}_{P}R\ {}^{P}_{L}\hat{R}\ [{}^{L}p_{\times}]\right] \\ &= -\left[\begin{bmatrix} s_\psi, c_\psi, \theta s_\psi \end{bmatrix}, \begin{bmatrix} s_\psi, c_\psi, \theta s_\psi \end{bmatrix} {}^{P}_{L}\hat{R}\ [{}^{L}p_{\times}]\right] \end{aligned} \tag{3.18}$$

For all rows in each iteration, ${}^{E}_{P}R$ is a constant full rank matrix; therefore, its actual value does not matter.

A new row of both $\mathbf{H}_H$ and $\mathbf{H}_V$ is added to $\mathbf{H}$ for each point on each patch which includes many scans. From the structure of $\mathbf{H}_H$ and $\mathbf{H}_V$ in eqns. (3.17) and (3.18) it is clear that changing θ and ψ will cause columns 1-3 to be linearly independent, and also causes columns 4-6 to be linearly independent. The variation in ${}^{L}p$ for different patches causes columns 1-3 to be linearly independent of columns 4-6.

Therefore, as long as the data for calibration includes at least two nearly orthogonal planes, with distinct and separated patches, that are scanned from poses with varying yaw and pitch angles, the matrix $\mathbf{H}$ will be full rank. Because $\delta\mathbf{x}$ is a constant vector (i.e. $\mathbf{A} = 0$ in the notation of [34]), this is equivalent to the system being observable.

3.1.3.5 Combined Plane Estimation for LIDAR Calibration

The analysis of the previous sections has considered the utilization of each LIDAR's data separately. Let q represent the number of planar patches in $P = S_H \cup S_V$. The method described above provides optimal LIDAR 1 calibration parameters x_1 and plane parameters $\{({}_1N_f, {}_1d_f) \text{ for } f = 1, \ldots, q\}$, using the data only from LIDAR 1. Similarly, the method provides optimal LIDAR 2 calibration parameters x_2 and plane parameters $\{({}_2N_f, {}_2d_f) \text{ for } f = 1, \ldots, q\}$, using the data only from LIDAR 2. This provides two estimates of the parameters for each plane, each optimal for the data set that was used.

To further refine the calibration, the data from the two LIDARs are combined for the purposes of plane fitting. For LIDAR ℓ, where $\ell = 1, 2$, using the method discussed in Section 3.1.3.3, let P_ℓ^f represent the set of points used for plane fitting and P_ℓ^r represent the set of points used for residual calculation for patch P. In this step, the plane parameters are estimated using $P_1^f \cup P_2^f$. Then, P_1^r is used to estimate LIDAR 1 calibration parameters x_1; and, P_2^r is used to estimate LIDAR 2 calibration parameters x_2. In each case, the rank of matrix $\mathbf{H}$ does not change, but the calibration accuracy improves with the two LIDAR's being forced to use the same estimates of the plane parameters for each patch.

3.1.4 Experimental Results

This section discusses the experimental design and results. Figs. 3.3 and 3.4 depict the data collection process.

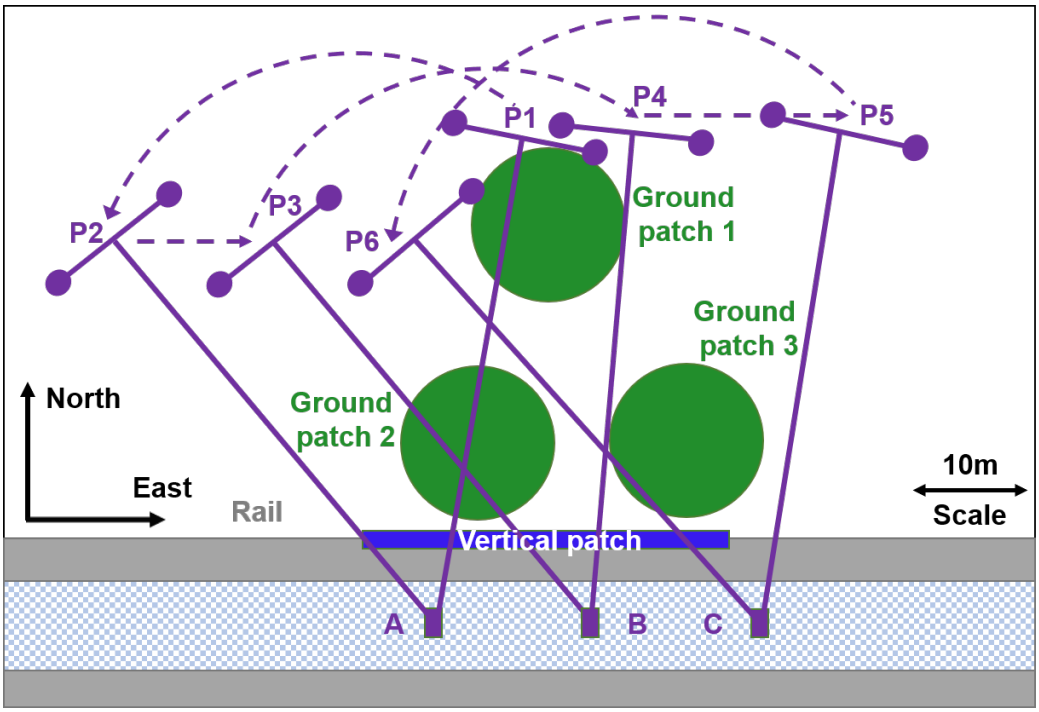


Figure 3.3: Platform pose during data collection and location of the planar patches.

3.1.4.1 Experiment Design

The sequence of boom orientations is illustrated in Fig. 3.3. The cross-hatched area at the bottom between the two wide gray lines represents the elevated structure on which the rail is mounted. The base moves along this rail. The purple rectangles on the rail represent three distinct base positions used during data collection. These are marked by labels (i.e., A, B, C). The two wide gray lines represent the vertical walls. Only one of the walls is scanned as the platform is always north of the wall. The solid purple lines represent the sequence of boom orientations at different time instants. Specific time instants are labeled (i.e., $P1,\ P2, \cdots, P6$) in both figures for cross-referencing. The dashed purple lines represent platform motion between two such instants. The green circular regions represents

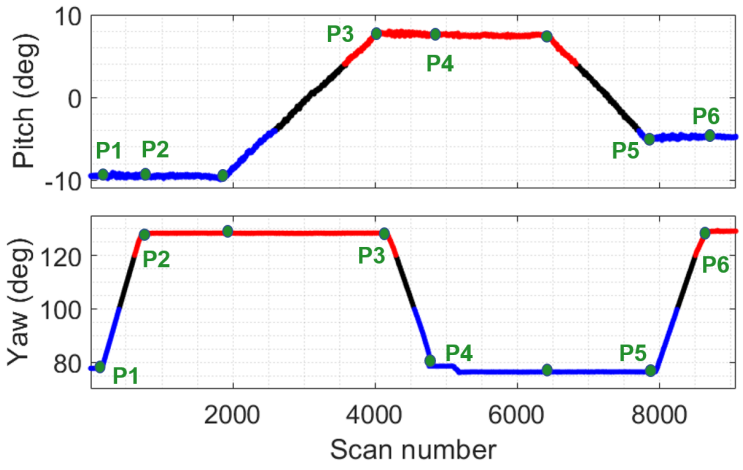


Figure 3.4: Platform pitch and yaw versus LIDAR scan number. Blue points indicate the pose data that are used to estimate the plane parameters. Red points show the pose data that are used to compute the residuals.

the horizontal plane patches. The portion of the rail wall that was scanned and used as a vertical patch is represented by the blue rectangle. Approximate orientation and scale are shown along the bottom of the figure. The platform pitch and yaw is plotted in Fig. 3.4.

For an empty portion of the stock yard, data collection proceeded as follows:

1. Data collection started with the base at position A, $\psi \approx 80°$ and $\theta \approx -9°$. This is $P1$.

2. Keeping the base position and pitch constant, the platform moved from $P1$ to $P2$ by changing ψ to approximately $130°$.

3. The base was translated to position B while holding ψ and θ constant. This scan number is indicated by the unlabelled green dot in Fig. 3.4 that is between $P2$ and $P3$.

4. The pitch was changed to $\theta \approx 8°$ keeping the base location and yaw constant. This

pose is marked as $P3$.

The rest of the data collection is continued in a similar pattern (see Figs. 3.3 and 3.4).

From this dataset four planar patches were extracted: three from the horizontal (ground) plane and one from the vertical (wall) plane. The patches are about 10 m in radius with centers separated by about the same amount. The scan data for each patch includes data scanned from numerous yaw angles and at least two pitch angles (separated as much as the platform structure allows). This ensures that the requirements of both Sections 3.1.3.3 and 3.1.3.4 are satisfied. The condition number of the resulting H matrix was 1954.

3.1.4.2 Results: Data Segmentation

The plane point extraction (see Section 3.1.3.1) process is straightforward. The decomposition of each extracted plane into patches is also straightforward (see Section 3.1.3.2). Neither will be discussed further herein.

After the plane extraction and clustering into patches, data in each patch P was split, based on the platform pose, to yield the point sets P^f for plane parameter extraction and P^r for extrinsic parameter estimation. The resulting split is indicated by the red and blue points in Fig. 3.4. For the ground plane patches, data was segmented based on the platform pitch. For the vertical plane patch, data was segmented based on the platform yaw.

Table 3.1: LIDAR 1 vertical plane and extrinsic parameter estimation

Iteration	**Estimated Normal**	**Estimated Distance (m)**	**Estimated Translation (m)**	**Estimated Rotation**
0	$[-0.027583,\ 0.995314,\ 0.092679]$	73.12	$[0.000,\ 0.307,\ -0.425]$	$[0°,\ 0°,\ 0°]$
1	$[-0.018278,\ 0.999526,\ 0.024781]$	81.10	$[-0.013,\ 0.289,\ -0.420]$	$[0.76°,\ 0.00°,\ -0.76°]$
2	$[-0.017460,\ 0.999771,\ 0.012408]$	81.81	$[-0.013,\ 0.289,\ -0.420]$	$[1.51°,\ 0.27°,\ -0.75°]$
3	$[-0.017460,\ 0.999771,\ 0.012408]$	81.81	$[-0.013,\ 0.289,\ -0.420]$	$[1.51°,\ 0.27°,\ -0.75°]$

Table 3.2: LIDAR 2 vertical plane and extrinsic parameter estimation

Iteration	**Estimated Normal**	**Estimated Distance (m)**	**Estimated Translation (m)**	**Estimated Rotation**
0	$[-0.011948,\ 0.999863,\ 0.011464]$	86.55	$[0.000,\ 0.277,\ 0.420]$	$[0°,\ 0°,\ 180°]$
1	$[-0.013170,\ 0.999724,\ 0.019479]$	85.51	$[0.000,\ -0.286,\ -0.425]$	$[-0.25°,\ 0.25°,\ 179.24°]$
2	$[-0.013300,\ 0.999720,\ 0.019574]$	85.40	$[0.001,\ -0.286,\ -0.425]$	$[-0.51°,\ 0.25°,\ 179.24°]$

Table 3.3: Combined extrinsic parameter estimation

Iteraion	**LIDAR 1**		**LIDAR 2**	
	Translation (m)	**Rotation**	**Translation (m)**	**Rotation**
1	$[-0.013,\ 0.289,\ -0.420]$	$[1.51°,\ 0.27°,\ -0.75°]$	$[0.001,\ -0.286,\ -0.425]$	$[-0.51°,\ 0.25°,\ 179.24°]$
2	$[-0.033,\ 0.259,\ -0.420]$	$[1.76°,\ 0.53°,\ -0.50°]$	$[0.031,\ -0.256,\ -0.425]$	$[-1.01°,\ 0.74°,\ 178.98°]$
3	$[-0.003,\ 0.279,\ -0.420]$	$[2.27°,\ 0.03°,\ -0.50°]$	$[0.031,\ -0.286,\ -0.425]$	$[-1.01°,\ 0.24°,\ 178.98°]$
4	$[-0.003,\ 0.279,\ -0.420]$	$[2.52°,\ 0.03°,\ -0.24°]$	$[0.031,\ -0.276,\ -0.425]$	$[-1.01°,\ 0.24°,\ 178.98°]$
5	$[-0.003,\ 0.279,\ -0.420]$	$[2.52°,\ 0.03°,\ -0.24°]$	$[0.031,\ -0.276,\ -0.425]$	$[-1.01°,\ 0.24°,\ 178.98°]$

Table 3.4: Plane parameter estimation in E-frame by the LIDARs

Iter-ation	$^{\mathbf{E}}\hat{\mathbf{N}}_{\mathbf{H_1}}$	$^{\mathbf{E}}\hat{\mathbf{d}}_{\mathbf{H_1}}$	$^{\mathbf{E}}\hat{\mathbf{N}}_{\mathbf{H_2}}$	$^{\mathbf{E}}\hat{\mathbf{d}}_{\mathbf{H_2}}$	$^{\mathbf{E}}\hat{\mathbf{N}}_{\mathbf{H_3}}$	$^{\mathbf{E}}\hat{\mathbf{d}}_{\mathbf{H_3}}$
1	$[-0.000600,\ 0.003440,\ 0.999994]$	6.03	$[-0.008563,\ 0.000411,\ 0.999963]$	7.13	$[-0.008090, -0.005507,\ 0.999952]$	7.30
2	$[-0.000731, -0.002732,\ 0.999996]$	5.10	$[\ 0.000521, -0.002335,\ 0.999997]$	6.25	$[\ 0.001270, -0.008904,\ 0.999960]$	6.26
3	$[-0.001652,\ 0.005474,\ 0.999984]$	5.40	$[-0.001492,\ 0.005588,\ 0.999983]$	5.58	$[-0.001140,\ 0.000737,\ 0.999999]$	5.41
4	$[-0.001954,\ 0.005429,\ 0.999983]$	5.13	$[-0.001643,\ 0.005503,\ 0.999984]$	5.44	$[-0.001301,\ 0.000918,\ 0.999999]$	5.29
5	$[-0.001954,\ 0.005429,\ 0.999983]$	5.13	$[-0.001643,\ 0.005503,\ 0.999984]$	5.44	$[-0.001301,\ 0.000918,\ 0.999999]$	5.29

Table 3.5: Plane parameter estimation in E-frame by the LIDARs

Iteration	${}^{\mathbf{E}}\hat{\mathbf{N}}_{\mathbf{V}}$	${}^{\mathbf{E}}\hat{\mathbf{d}}_{\mathbf{V}}$
1	$[-0.015270,\ 0.998281,\ 0.056581]$	83.83
2	$[-0.017958,\ 0.999155,\ 0.036971]$	81.50
3	$[-0.015309,\ 0.999683,\ 0.020012]$	83.68
4	$[-0.016139,\ 0.999561,\ 0.024846]$	82.98
5	$[-0.016139,\ 0.999561,\ 0.024846]$	82.98

3.1.4.3 Results: Calibration

For LIDAR 1 and LIDAR 2, Tables 3.1 and 3.2 show the estimated vertical plane normal, distance and extrinsic calibration parameter after each iteration of the optimization. While the normals estimated by the two LIDARs differ by only 8.2 milliradians, the two LIDARs estimates of the d parameter (i.e., 81.81 and 85.40) differ by 3.59 m, which is large relative to the desired calibration accuracy and LIDAR range standard deviation. The explanation of this is straightforward. Rotation of any vector $\mathbf{v}$ about an arbitrary axis $\mathbf{r}$ by an angle θ is given by the following transformation (see Section 9.2.4 in [13]):

$$T(\mathbf{v}) = \left[(1-\cos\theta)\,\mathbf{r}\mathbf{r}^{\top} + \cos\theta\mathbf{I} + \sin\theta\,[\mathbf{r}_{\times}]\right]\mathbf{v}.$$

If θ is small such that small angle approximation is valid, the above equation simplifies to

$$T(\mathbf{v}) = \begin{bmatrix} 1 & -\theta r_z & \theta r_y \\ \theta r_z & 1 & -\theta r_x \\ -\theta r_y & \theta r_x & 1 \end{bmatrix} \mathbf{v}$$

$$= (I + \theta [\mathbf{r}_\times]) \mathbf{v}.$$

Because the normal vector has unit length, the error model relating true normal N and estimated normal $\hat{N}$ can be written as

$$\hat{N} = (I + \theta_1 [\mathbf{u_1}_\times] + \theta_2 [\mathbf{u_2}_\times]) N$$

where $\mathbf{u_1}$ and $\mathbf{u_2}$ are unit vectors orthogonal to N and each other, and θ_1 and θ_2 are the small error angles around $\mathbf{u_1}$ and $\mathbf{u_2}$. If p_0 is a point on the plane, then $\mathbf{u_1}$ and $\mathbf{u_2}$ may be defined as

$$\mathbf{u_1} = \frac{p_0 - (p_0 \cdot N) N}{\| p_0 - (p_0 \cdot N) N \|} \text{ and } \mathbf{u_2} = N \times \mathbf{u_1}$$

The $E-$frame origin is defined by the location of the GNSS base station antenna. For this experiment, the dataset used to estimate this plane for patch P has a centroid $p_c = [863.40, 96.78, 7.04]$. The computed values of $\mathbf{u_1}$ and $\mathbf{u_2}$ are $[0.999824,\ 0.017374,\ 0.006965]$ and $[0.006748,\ 0.012527,\ -0.999899]$, respectively when $p_0 = p_c$. Note that, according to this definition of the axes, $N \times \mathbf{u_1} = \mathbf{u_2}$, $\mathbf{u_1} \times \mathbf{u_2} = N$ and $\mathbf{u_2} \times N = \mathbf{u_1}$. The estimated

and actual d values are related as:

$$\begin{aligned}
\hat{d} &= \hat{N} \cdot p_0 \\
&= \left(I + \theta_1 \left[\mathbf{u_1}_\times\right] + \theta_2 \left[\mathbf{u_2}_\times\right]\right) N \cdot p_0 \\
&= \left(N + \theta_1 \left(\mathbf{u_1} \times N\right) + \theta_2 \left(\mathbf{u_2} \times N\right)\right) \cdot p_0 \\
&= \left(N - \theta_1 \mathbf{u_2} + \theta_2 \mathbf{u_1}\right) \cdot p_0 \\
&= d - \theta_1 \mathbf{u_2} \cdot p_0 + \theta_2 \mathbf{u_1} \cdot p_0 \qquad (3.19) \\
&= d + 0.001\theta_1 + 864.979\theta_2. \qquad (3.20)
\end{aligned}$$

The parameter d is the distance from the E-frame origin to the closest point on the corresponding plane. For $p_0 \in P$, where p_0 is within $L = 10$ m of p_c, (3.20) shows that each milli-radian of angular error (i.e. θ_1 or θ_2) can result in $d - \hat{d}$ being inaccurate by 0.865 m, due to $\mathbf{u_1} \cdot p_0$ being large. Therefore, the 8.2 milli-radian by which the two normal estimates differ easily explains the 3.59 m by which the estimates of d differ.

The residual for a point P_i that is on the plane is

$$\begin{aligned}
r_{p_i} &= \hat{N} \cdot p_i - \hat{d} \\
&= d - \theta_1 \mathbf{u_2} \cdot p_i + \theta_2 \mathbf{u_1} \cdot p_i - \left(d - \theta_1 \mathbf{u_2} \cdot p_0 + \theta_2 \mathbf{u_1} \cdot p_0\right) \\
&= \left(\theta_2 \mathbf{u_1} - \theta_1 \mathbf{u_2}\right) \cdot \left(p_i - p_0\right).
\end{aligned}$$

By the definition of P, $\|p_i - p_c\| \leq L = 10$ m; therefore, the effect of the normal error on the residual is on the order of 1.5 cm which is less than the standard deviation of the LIDAR range or the residual standard deviation as shown in Fig. 3.5.

Table 3.3 shows the extrinsic calibration parameters of both LIDARs after each iteration when using the combined data as described in Section 3.1.3.5. Norm of the vec-

tor $\delta\mathbf{x}$ was used ($\delta\mathbf{x} < 0.001$) for terminating the iteration. Table 3.4 and 3.5 show the estimated plane parameters for the horizontal patches and the vertical patch using data from both LIDARs. The three plane patches represented in Table 3.4 are portions of a paved construction intended to be planar. The maximum angular deviation between the estimated normals at iteration 5 is 4.6 milli-radian. By the analysis ending with (3.19) this angular deviation in the normals explains the 0.31 m difference in the estimated values of d.

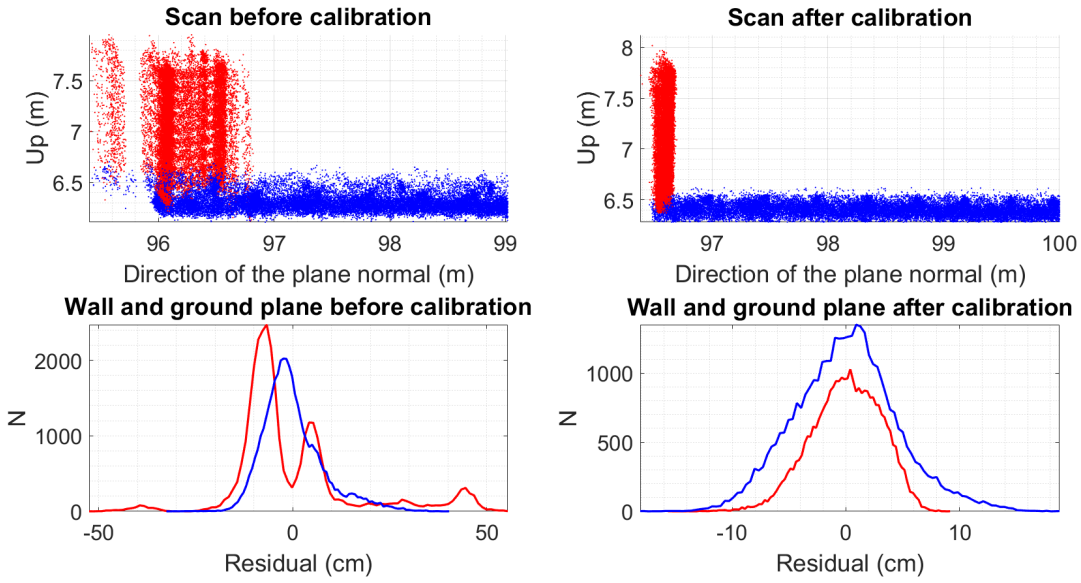


Figure 3.5: Georectified points from a BWR ground plane patch (blue) and wall (red) plane patch before (left) and after (right) calibration.

Fig. 3.5 shows georectifed data ${}^{E}\hat{p}_i$ for a section of the wall and ground plane near the wall before the start of calibration (left) and after calibration using the combined data from both LIDARs (right). The viewing angle is parallel to the vector $\hat{N}_V \times \hat{N}_H$, which is orthogonal to both the estimated wall and ground planes. Before calibration, the point

cloud generated from the vertical (i.e., wall) plane extends (visually) over a full meter, with the residual having a standard deviation of 15.84 cm, while the histogram is multimodal. Similarly, ground plane residual has a standard deviation of 7.84 cm and a non-Gaussian histogram. After calibration, the residuals for both patches are much closer to Gaussian and the wall and ground plane patches have standard deviations that have reduced to 3.06 and 4.55 cm, respectively, which are near the manufacturer's specifications for the range measurement standard deviations as stated in Footnote 2.

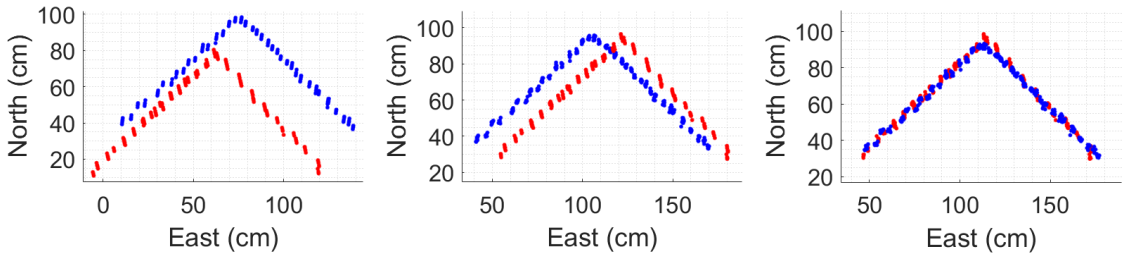


Figure 3.6: Scan of a test box initially and after both stages of calibration (separate and combined LIDAR data). The blue and red points represent points scanned by LIDAR 1 (blue) and LIDAR 2 (red), respectively. The figure on the left shows the (mis)aligned scans before calibration. The figure in the middle shows alignment after calibration using each LIDAR's own data separately (see Section 3.1.3.4). The figure on the right shows the alignment after calibration using the combined data (see Section 3.1.3.5).

A further test was performed to evaluate calibration accuracy. During the data collection, a small box was placed on the ground in the area that was scanned. Due to the relative position of the box and the scanners, only two sides of the box were scanned. The points from those two sides of the box were extracted using the incremental line extraction approach. This data was not used during calibration. Fig. 3.6 shows the georectified box points as computed for LIDAR 1 (blue) and LIDAR 2 (red). Two planes were fitted, one

on each side of the box using data from each LIDAR separately. The planes were compared between the LIDARs. Before calibration, the average distance between the scanned planar regions were 26.27 cm and 8.66 cm respectively. After calibration using each LIDAR's data separately, the average distance between the planes were 8.35 cm and 13.17 cm respectively. After combining the data from both LIDARs, the average distance between the planes were 2.00 cm and 0.58 cm respectively. Therefore, subdecimeter relative accuracy is achieved for both planes.

3.2 GNSS to Gantry Extrinsic Calibration

This section discusses the GNSS to Gantry extrinsic calibration estimation which is required to compute the gantry position from the platform frame position.

3.2.1 Problem Statement

The PLC controlling the BWR moves the wheel to a desired location by changing the position of the gantry on the track, yaw, and/or pitch of the arm. The yaw and pitch estimate is obtained from the GNSS position estimates. Since there is no sensor providing the gantry position information, it needs to be computed from the platform pose provided by the GNSS receivers. The current gantry position ${}^{E}p_g$, platform position ${}^{E}p_p$, platform orientation ${}^{E}_{P}R$ and the calibration parameter ${}^{P}T_{pg}$ are related by the following equation:

$$ {}^{E}p_g \;=\; {}^{E}p_p \;+\; {}^{E}_{P}R\;{}^{P}T_{pg} \tag{3.21} $$

The objective is to estimate T_{pg} so that the gantry position can be estimated in real-time using eqn. (3.21).

3.2.2 Solution Approach

The platform to gantry vector T_{pg} can be estimated by collecting GNSS data while the gantry is stationary. Eqn. (3.21) yields,

$$ {}^{E}p_p \;=\; {}^{E}p_g \;-\; {}^{E}_{P}R\ {}^{P}T_{pg} \tag{3.22} $$

$$ {}^{E}p_p \;= \begin{bmatrix} I & -{}^{E}_{P}R \end{bmatrix} \begin{bmatrix} {}^{E}p_g \\ {}^{P}T_{pg} \end{bmatrix} \tag{3.23} $$

If the platform pose is varied by changing the yaw and pitch of the platform while keeping the gantry position ${}^{E}p_g$ fixed, we get the following system of equations

$$ \begin{bmatrix} {}^{E}p_{p_1} \\ {}^{E}p_{p_2} \\ {}^{E}p_{p_3} \\ \vdots \\ {}^{E}p_{p_m} \end{bmatrix} = \begin{bmatrix} I & -{}^{E}_{P}R_1 \\ I & -{}^{E}_{P}R_2 \\ I & -{}^{E}_{P}R_3 \\ \vdots & \\ I & -{}^{E}_{P}R_m \end{bmatrix} \begin{bmatrix} {}^{E}p_g \\ {}^{P}T_{pg} \end{bmatrix} \tag{3.24} $$

If multiple sets of similar data is collected from different gantry positions, the system of equations become

$$ \begin{bmatrix} {}^{E}p_{p_1} \\ {}^{E}p_{p_2} \\ {}^{E}p_{p_3} \\ \vdots \\ {}^{E}p_{p_m} \end{bmatrix} = \begin{bmatrix} I & 0 & \cdots & 0 & -{}^{E}_{P}R_1 \\ 0 & I & \cdots & 0 & -{}^{E}_{P}R_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & I & -{}^{E}_{P}R_m \end{bmatrix} \begin{bmatrix} {}^{E}p_{g_1} \\ {}^{E}p_{g_2} \\ \vdots \\ {}^{E}p_{g_n} \\ {}^{P}T_{pg} \end{bmatrix} \tag{3.25} $$

which has the form $y = HX$ and can be solved for X numerically. Since we are interested in the vector platform to gantry ${}^{P}T_{pg}$, not the whole X, we can solve a smaller problem which

computes ${}^{P}T_{pg}$ only using Schur decomposition. Eqn. (3.25) can be written in following form:

$$y = HX \tag{3.26}$$

$$H^{\top} y = H^{\top} HX \tag{3.27}$$

$$z = MX \tag{3.28}$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} X_g \\ {}^{P}T_{pg} \end{bmatrix} \qquad \text{using Schur decomposition} \tag{3.29}$$

where $A \in R^{3\times 3n}$, $B \in R^{3\times 3}$, $C \in R^{3n\times 3n}$, $D \in R^{3n\times 3}$, $X_g \in R^{3n\times 1}$ and ${}^{P}T_{pg} \in R^{3\times 1}$ Therefore, ${}^{P}T_{pg}$ can be found by solving the following system of equations:

$$z1 - AC^{-1} z_2 = \left(B - AC^{-1} D\right) {}^{P}T_{pg} \tag{3.30}$$

3.2.3 Experimental Results

A set of data was collected keeping the gantry position constant and changing the platform orientation only. Then the gantry was moved and another set of data was collected and so on. After accumulating multiple data sets, eqn. (3.30) was solved to estimate the platform to gantry calibration parameter. The estimated value was ${}^{P}T_{pg} = [-45.76, \ -7.39, \ -2.87]$ m.

3.3 Conclusion

This chapter presented a calibration method for multi-LIDAR BWR's that is performed in the standard BWR environment without using special calibration targets or

special measurement markers. The approach uses two planar features (i.e., the vertical wall that supports the BWR rail and the ground planes) which are both readily available in all BWR systems. The method and its accuracy have been demonstrated using experimental data from a port located in Yantai, China. Sub-decimeter accuracy of approximately 2 cm is achieved.

This chapter also discusses the calibration between the platform and gantry using GNSS data which is required to estimate gantry position in real-time.

Chapter 4

Surface Reconstruction

The chapter investigates the problem of real-time point cloud management and visualization for stockyards environments.

4.1 Literature Review

Based on a review of several major iron-ore facilities, it was found that the stockpile materials are currently being reclaimed at approximately 50% of their potential engineering production rates [30]. Therefore automation is of crucial importance. Real-time point cloud management is a pre-requisite for real-time visualization, surface feature extraction, and automation. The high acquisition rate and need for high spatial resolution pose considerable challenges for point cloud management, visualization, and feature extraction.

Lu presents the groundwork for treating a BWR as a robotic arm [29]. It focuses on BWR's using encoder-based positioning and does not address point cloud management. Zhao et al.[49, 50] proposed a 3D volumetric model able to dynamically represent the

stockpile. The model demonstrated a high degree of accuracy; however, changes in the surface require a new model to be fitted and computation time for each fitting averages about $81s$. Zhao et al.[51] propose a voxel model to represent and compute the volume of the surface. The algorithm takes about 80s to generate the model for a stockpile with 155m length and 45m width. Additionally, none of the above work addresses the issue of real-time visualization of the surface which is crucial for remote BWR operation. Point cloud triangulation which is required to display solid surfaces is also an active research problem[33]. Su et al. proposed a method for rapid triangulation using adaptive Hilbert curves [39]. However, it is still computationally too expensive to implement in real-time.

LIDAR based SLAM techniques [27, 46, 52] build a map of the environment using point cloud data. Kohlbrecher et al. [27] use occupancy grid maps [41] to store the map of the environment using down-sampled LIDAR points. Zhang et al. use a 3D KD-tree to store the point cloud. The environment is assumed to be static; therefore, these methods do not fit our application. Zlot et al. [52] proposed a method for large scale 3D mapping using LIDAR data. Because it uses batch processing it is not suitable for BWR applications.

4.2 Point Cloud Management

Point cloud management refers to the problem of managing the georectified points as they arrive from the sensors. Note that, the surface is time-varying as the reclaimer wheel changes its shape while digging. So, instead of just accumulating the point cloud, it needs to be maintained. New points have to be added and obsolete points must be removed.

Each LIDAR generates points at a high rate. For a LIDAR rotating 15 revolutions

per second with measurements each 0.25° there are 1440 points per rotation, 21600 points per LIDAR per second, and 43200 points per second total. This data must be managed efficiently for real-time operation.

A uniform grid spatial partitioning technique [15] overlays a regular grid over the stockyard. Each georectified point is mapped to the cell that contains it. Due to the uniformity of the grid, accessing a cell corresponding to a particular point is both simple and fast: the east and north values are simply divided by the cell size to obtain cell indices. The cell size influences the maximum resolution, memory and computational requirements. A grid size of 5 cm is used herein to achieves the user-desired accuracy without loosing its real-time capability.

To manage the point cloud efficiently in real-time, two types of data structures are used in the implemented software, one for the point cloud and one for the image. The LIDAR does not provide data for all cells of the grid; therefore, some grid cells may be empty or contain old data. Therefore, the point cloud in its raw form is not suitable for visualization. The second data structure is a matrix of real numbers (i.e., a raster image) that is used for the user display. It cannot contain any holes.

4.2.1 Point Cloud

The point cloud data is stored in a matrix of linked lists, one for each cell of the grid. The linked list for each cell contains the list of LIDAR returns in that cell or may be empty. As the platform moves, new georectified points are added to the linked lists for the appropriate grid cells. However, as new points are added to the cells, other cells containing

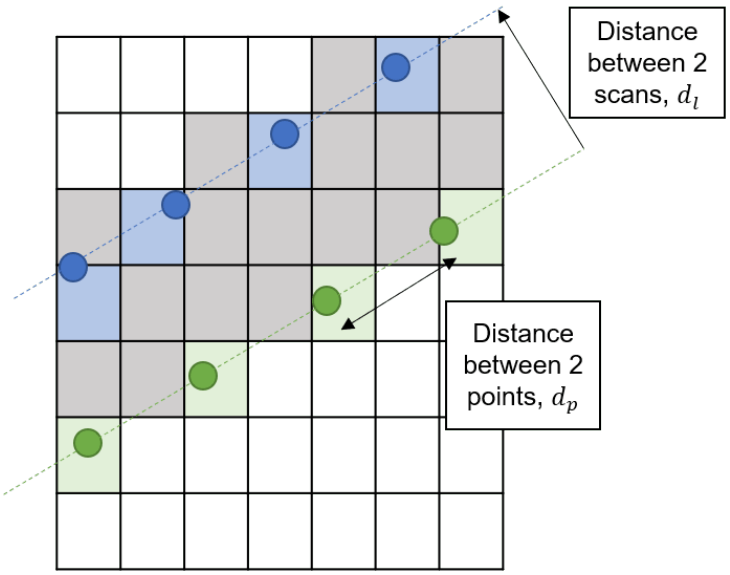


Figure 4.1: Two consecutive LIDAR scans (green and blue circles along green and red lines). The green and blue shaded squares represent cells containing new points. The gray squares represent cells potentially made obsolete by the two consecutive LIDAR scans.

old points become obsolete which need to be deleted and removed from visualization.

Fig. 4.1 shows a grid and two georectified LIDAR scan lines. The green circles represent points from scan line l_k measured at time t_k. The green squares represent the cells where points are added. The blue circles represent points from scan line l_{k+1} measured at time t_{k+1} and the blue squares represent the cells where these points are added. The separation d_p between two consecutive points on a single scan is related to the LIDAR angular resolution $\delta\alpha$ as $d_p = R\delta\alpha$ (assuming a planar surface perpendicular to the beam). The distance d_p increases with R and with the angle of incidence. The distance d_l between two such consecutive georectified scan lines depends on the platform yaw rate and the scan rate of the LIDAR. Many cells along a given scan line and many cells between consecutive scan lines are missed, not receiving new data. In addition, any cells along and in the region

between the two scan lines may contain old data that may no longer represent the surface. Obsolete data in these cells must be deleted. Adding new points and deleting old points is accomplished by a linked list manager that only operates on the cells between the two scan lines. This linked list manager adds the indices of cells that it has changed onto FIFO buffer that is shared with the raster image manager.

4.2.2 Raster Matrix

A raster image I of the surface is constructed for visualization from the point cloud matrix of linked lists. The resolution of I is a tunable parameter involving trade-offs between computational load and image quality as a function of resolution. The raster I is a 2D matrix of real values, where the value of each matrix element is the current height of the corresponding cell. For visualization, the raster matrix cannot have any undefined elements (i.e., holes).

The raster image manager uses and removes elements from the FIFO buffer. Its job is to try to empty that buffer, while adjusting the raster matrix elements to keep them accurate. New data updates the corresponding matrix element. Removal of obsolete data from the point cloud, leaving a hole in the point cloud, is fixed in the raster image by interpolating between nearby valid data. Thus I always maintains a raster image of the point cloud that contains structured points that is easy to use for visualization.

4.3 Visualization

This section discusses the real-time visualization of the surface using the raster image I. This process uses triangulation, which is required to draw a solid surface instead of discrete points with normal and color computation of each triangle as is required for proper lighting and display of the triangles.

4.3.1 Triangulation

Triangulation is a important step of all reconstruction problems [14]. A triangulation converts a given set of points into a consistent polygonal model (mesh). The input for the triangulation are the points of the image I. Delaunay Triangulation (DT) simultaneously optimizes several quality measures e.g. angles, edge lengths, height and area of the triangles [20].

However, triangulation algorithms are expensive, so it is not feasible to update the triangulation in real-time as the points in the point cloud are updated. Instead, the triangulation is performed once over the fixed structure of I. The geographic extent of each element of I is small enough that the exact position of LIDAR returns in a cell do not affect the image quality. First, a DT is computed assuming all image cells have a point at the center. Then this DT is kept unchanged in the future. The use of fixed DT greatly reduces computational load.

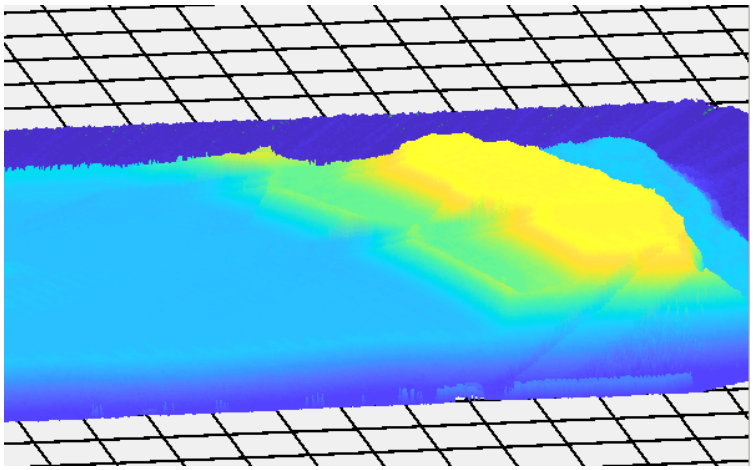
4.3.2 Normal and Color Estimation

Drawing the surface requires computation of the surface normal so that the graphics library can determine how light reflects from the surface. Given the vertices p_0, p_1 and p_2 of a triangle, the normal n is computed as:

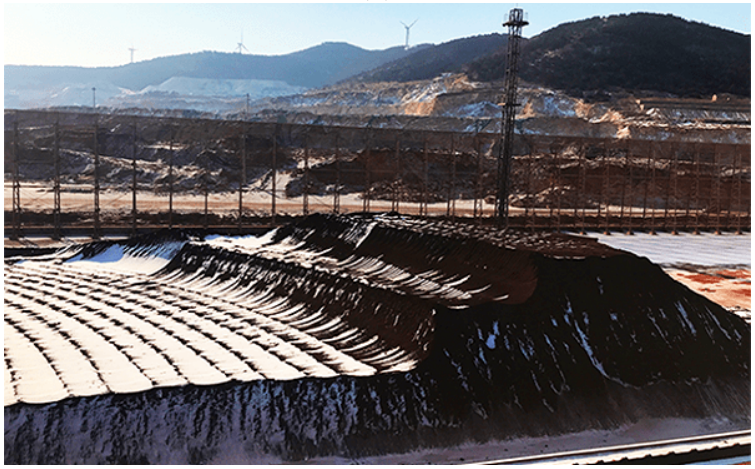
$$u = p_1 - p_0,\ v = p_2 - p_0,\ \text{and}\ n = u \times v.$$

The normal n is then assigned to vertices p_0, p_1, and p_2. When a pixel is updated on the raster image I, the normals of the triangles that the pixel belongs to, need to be updated. Therefore, each time a pixel is updated, the normals are also updated. Each triangle vertex is assigned a color based on its height.

Fig. 4.2(a) shows the surface drawn in real-time using BWR data. Fig. 4.2(b) shows an image of the actual pile.

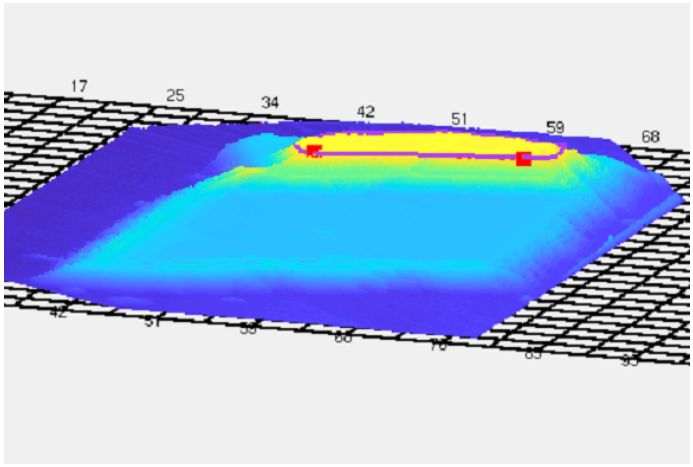


(a)



(b)

Figure 4.2: (a) Real-time surface visualization. The materials are occupying an area of 200m $\times$ 40m. The total number of points are 2M. (b) Image of the actual pile.



(a)

(b)

Figure 4.3: (a) Surface contour detection and feature extraction. (b) Image of the actual surface.

Chapter 5

Automatic Control

This section discusses the automatic computation of BWR entry and exit points and material volume of stockpile.

5.1 Entry and Exit Point Selection

The reclaimer scoops material from the surface layer-by-layer from top-to-bottom. Feature extraction refers to the step of calculating the entry and exit point for the next sweep of the boom using the surface data. This step is performed using the following steps:

1. *Contour detection:* After finding the highest point z^*, the vicinity of z^* is scanned to construct the closed contour C around z^* such that: (a) the interior of C is connected; (b) all points in the interior of C have $z \geq z^* - h$; (c) all cells on C have $z < z^* - h$. C is an unordered set of cells which is processed to remove holes from its interior and to sort the list such that c_{i-1} and c_i share an edge or corner for all i where c_{i-1} and

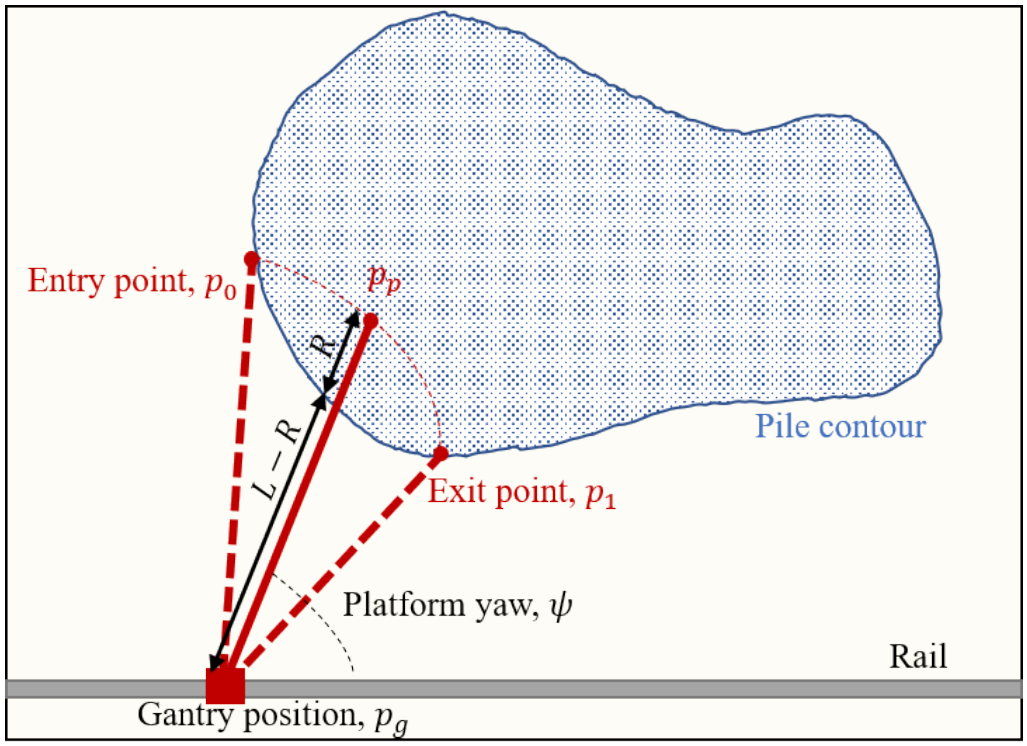


Figure 5.1: Entry and exit point detection.

c_i denote the $(i-1)$-th and i-th element of C respectively.

2. *Entry and exit point detection:* Given the contour C, the entry point p_0 and exit point p_1 can be computed. Refer to Fig. 5.1. The blue curve represents the pile contour and the red lines represent the boom at different platform yaw. The mid-point p_p during the sweep of the boom is related to the gantry position p_g and platform yaw ψ by

$$p_p = p_g + (L-R)\begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix} \tag{5.1}$$

where L is the length of the boom and R is the radius of the wheel. The gantry position p_g and the pile point p_p can be solved using eqn. (5.1) and optimizing any user defined

criteria e.g. maximizing scoop volume, minimizing x-position of the gantry etc. After p_g and p_p are computed, the entry point p_0 and exit point p_1 are computed as shown in Fig. 5.1.

Fig. 4.3(a) shows the surface with top layer contour detected (purple curve). The red squares represent the entry and exit point.

5.2 Volume Computation

Managing the point cloud in a grid also enables computing the volume of the surface easily. The volume of the stockpile is computed by adding the volume of each cell in the grid. Given the cell size of l, the volume v_{ij} of the cell s_{ij} is computed as

$$v_{ij} = l^2 w_{ij} \tag{5.2}$$

where w_{ij} is the height of the materials in the cell. As the ground of the stockyard is flat and thus can be accurately modeled by a plane with normal N and distance d, a set of ground points are stored by scanning the stockyard when it is empty. The plane normal N and distance d are estimated from the ground points using methods discussed in Section 3.1.3.3. Then given a cell location i and j, the height of the ground g_{ij} is computed from N and d. The height of the material is then $w_{ij} = h_{ij} - g_{ij}$ where h_{ij} is the height of cell s_{ij}. Whenever a cell is updated, the volume of the cell is also computed and updated.

5.3 Database Backup and Communication

The point cloud is stored in the volatile primary memory of the computer. To prevent accidental data loss and provide resume capability of the software, a copy of the point cloud is stored in a database as a backup. When a point is added or removed from the point cloud, its coordinate along with a unique ID is put in a first in first out (FIFO) buffer. The database is then continuously updated using the data in the buffer. This data is stored in the hard drive after the termination of the software. If the user wishes to resume the operation, during the start up of the software, the grid is populated using the data stored in the database. When the empty stockyard is scanned to collect data from the ground, those points are also stored in the database, this requires scanning the ground only once to extract the plane parameters.

Additionally, the software connects to a PLC to read and write data stored in its memory. The estimated platform pose, gantry position, wheel position, extracted entry and exit points are sent to the PLC for automatic control. However, this aspect is out of the scope of this dissertation.

Chapter 6

Real-time Implementation

This chapter discusses the software implementation for the automation.

The software is implemented as a multi-thread program written in *C++*. Fig. 6.1 shows the system configuration and structure of the software. The rectangles represent different child threads. The arrows represent data sharing between threads. All data transfers are performed via ring/circular buffers to prevent data loss. There are eight child threads under the main thread of the program that perform different steps. The four sensor threads (e.g. *GNSS 1*, *GNSS 2*, *LIDAR 1*, and *LIDAR 2*) parse the raw data coming from the

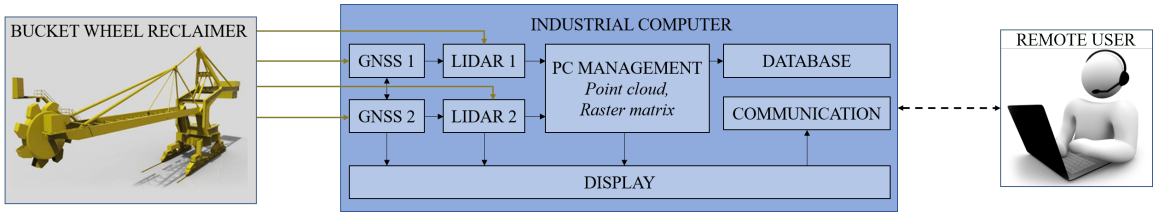


Figure 6.1: System configuration and the software architecture implemented in *C++*.

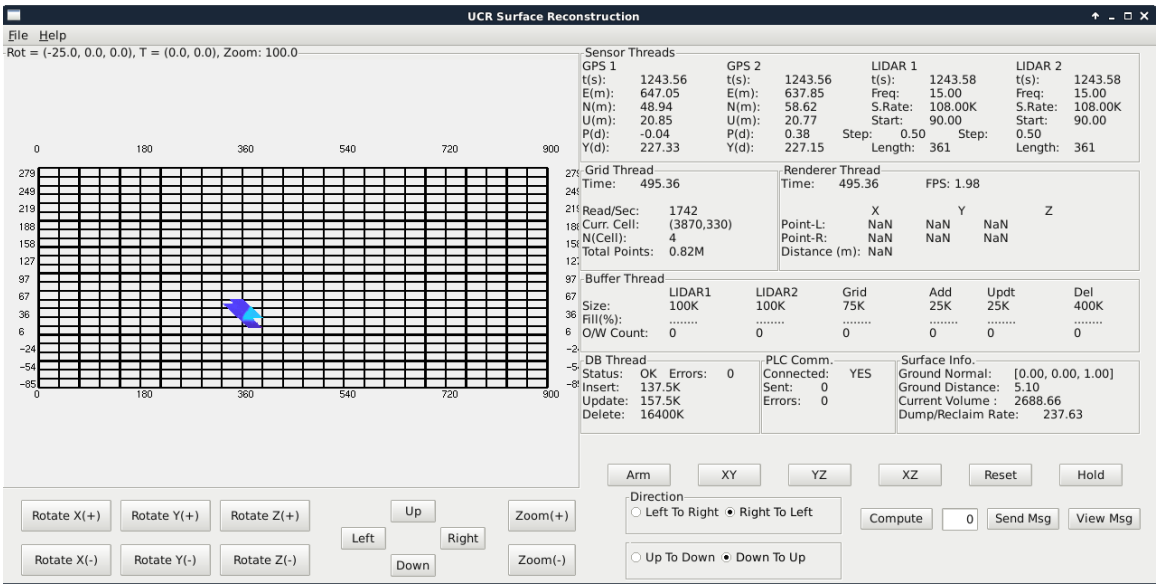


Figure 6.2: A snapshot of the real-time software during operations.

sensors and generate georectified LIDAR data. The *PC Management* thread receives the data and maintains two important variables in the memory, the point cloud and raster I as explained in Section 4.2. The *PC Management* thread also sends data to the *Database* thread which keeps a back-up copy of the point cloud in a SQL database. The *Display* thread displays the raster I using the *OpenGL* library [25]. The *Display* thread also sends data to the *Communication* thread that can send the extracted entry and exit points to a PLC. The graphical user interface (GUI) uses the *wxWidgets* library[38] and the matrix computations use the *Eigen* library[22].

Chapter 7

Conclusions and Future Work

This chapter lists the publications resulting from the BWR study and discusses the possible future studies.

7.1 Publications

1. M. Billah, J. A. Farrell, "Bucket Wheel Reclaimer Calibration", IEEE American Control Conference, (Accepted), 2019.

2. M. Billah, J. A. Farrell, "Bucket Wheel Reclaimer Extrinsic Parameter Calibration", in IEEE Transactions on Control Systems Technology, (Submitted), 2018.

3. M. Billah, J. A. Farrell, "Real-time Time-varying Surface Reconstruction and Edge Point Extraction For Bucket Wheel Reclaimers", IEEE Conference on Control Technology and Applications, (Accepted), 2019.

7.2 Future Work

There are a number of works that can be extended from the current studies. First, the calibration of the GNSS-LIDAR is now performed offline. This can be carried out in real-time.

Inertial sensors like 6 DOF IMU can be used for platform localization which will improve the platform position accuracy. Outlier detection algorithms can be implemented to detect spurious measurements.

The graphics computations can be implemented in the GPU to improve the quality of the on-screen visualization (e.g. resolution) and accelerate the frame rate.

Bibliography

[1] 2d lidar sensors ld-lrs. [Online; accessed 16-January-2019].

[2] Adequacy of Solutions.

[3] USER GUIDE - Trimble BD982 GNSS Receiver Module.

[4] Hatem Alismail and Brett Browning. Automatic calibration of spinning actuated lidar internal parameters. *J. of Field Robotics*, 32(5):723–747, 2015.

[5] Nina Amenta and Yong Joo Kil. Defining point-set surfaces. In *ACM T. on Graphics (TOG)*, volume 23, pages 264–270. ACM, 2004.

[6] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM T. on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

[7] M Billah and J.A. Farrell. Bucket wheel reclaimer extrinsic parameter calibration. *IEEE T. on Control System Technology*, Submitted.

[8] M Billah and J.A. Farrell. Technical note with supporting information for bucket wheel reclaimer extrinsic parameter calibration. *To be placed online later*, Submitted.

[9] Michael Bosse and Robert Zlot. Continuous 3d scan-matching with a spinning 2d laser. *IEEE Int. Conf. on Robotics and Automation*, pages 4312–4319, 2009.

[10] Michael Bosse, Robert Zlot, and Paul Flick. Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *IEEE T. on Robotics*, 28(5):1104–1119, 2012.

[11] Dong-Geol Choi, Yunsu Bok, Jun-Sik Kim, and In So Kweon. Extrinsic calibration of 2-d lidars using two orthogonal planes. *IEEE T. on Robotics*, 32(1):83–98, 2016.

[12] Clark E Cohen et al. Attitude determination. *Global Positioning System: Theory and applications.*, 2:519–538, 1996.

[13] Ian R Cole. *Modelling CPV*. PhD thesis, Loughborough Unversity, 2015.

[14] Herbert Edelsbrunner. *Geometry and topology for mesh generation*, volume 7. Cambridge University Press, 2001.

[15] Christer Ericson. *Real-time collision detection*. CRC Press, 2004.

[16] Jay Farrell. *Aided navigation: GPS with high rate sensors*. McGraw-Hill, Inc., 2008.

[17] Eduardo Fernández-Moral, Vicente Arévalo, and Javier González-Jiménez. Extrinsic calibration of a set of 2d laser rangefinders. *IEEE Int. Conf. on Robotics and Automation*, pages 2098–2104, 2015.

[18] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[19] David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[20] Steven Fortune. Voronoi diagrams and delaunay triangulations. In *Computing in Euclidean geometry*, pages 225–265. World Scientific, 1995.

[21] Chao Gao and John R Spletzer. On-line calibration of multiple lidars on a mobile vehicle platform. *IEEE Int. Conf. on Robotics and Automation*, pages 279–284, 2010.

[22] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. [Online; accessed 16-January-2019].

[23] Mengwen He, Huijing Zhao, Jinshi Cui, and Hongbin Zha. Calibration method for multiple 2d lidars system. *IEEE Int. Conf. on Robotics and Automation*, pages 3034–3041, 2014.

[24] Jaehyeon Kang and Nakju Lett Doh. Full-dof calibration of a rotating 2-d lidar with a simple plane measurement. *IEEE T. on Robotics*, 32(5):1245–1263, 2016.

[25] Khronos Group. Open graphics library (opengl). [Online; accessed 16-January-2019].

[26] Klaas Klasing, Daniel Althoff, Dirk Wollherr, and Martin Buss. Comparison of surface normal estimation methods for range sensing applications. *IEEE Int. Conf. on Robotics and Automation*, pages 3206–3211, 2009.

[27] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. *IEEE Int. Symposium on Safety, Security, and Rescue Robotics*, pages 155–160, 2011.

[28] Jesse Levinson and Sebastian Thrun. Unsupervised calibration for multi-beam lasers. In *Experimental Robotics*, pages 179–193. Springer, 2014.

[29] Tien-Fu Lu. Preparation for turning a bucket wheel reclaimer into a robotic arm. *IEEE Int. Conf. on Robotics and Biomimetics*, pages 1710–1715, 2009.

[30] Tien-Fu Lu and Maung Thi Rein Myo. Optimal stockpile voxel identification based on reclaimer minimum movement for target grade. *Int. J. of Mineral Processing*, 98(1-2):74–81, 2011.

[31] Naveed Muhammad and Simon Lacroix. Calibration of a rotating multi-beam lidar. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 5648–5653, 2010.

[32] Viet Nguyen, Stefan Gächter, Agostino Martinelli, Nicola Tomatis, and Roland Siegwart. A comparison of line extraction algorithms using 2d range data for indoor mobile robotics. *Autonomous Robots*, 23(2):97–111, 2007.

[33] Fabio Remondino. From point cloud to surface: the modeling and visualization problem. *Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34, 2003.

[34] Ihnsoek Rhee, Mamoun F Abdel-Hafez, and Jason L Speyer. Observability of an integrated gps/ins during maneuvers. *IEEE T. on Aerospace and Electronic Systems*, 40(2):526–535, 2004.

[35] GK Robinson. How much would a blending stockpile reduce variation? *Chemometrics and intelligent laboratory systems*, 74(1):121–133, 2004.

[36] Mark Sheehan, Alastair Harrison, and Paul Newman. Self-calibration for a 3d laser. *The Int. J. of Robotics Research*, 31(5):675–687, 2012.

[37] Ali Siadat, Axel Kaske, Siegfried Klausmann, Michel Dufaut, and René Husson. An optimized segmentation method for a 2d laser-scanner applied to mobile robot navigation. *IFAC symposium on intelligent components and instruments for control applications*, pages 153–158, 1997.

[38] Julian Smart et al. wxwidgets. [Online; accessed 16-January-2019].

[39] Tianyun Su, Wen Wang, Zhihan Lv, Wei Wu, and Xinfang Li. Rapid delaunay triangulation for randomly distributed point cloud data using adaptive hilbert curve. *Computers & Graphics*, 54:65–74, 2016.

[40] RM Taylor and Penny J Probert. Range finding and feature extraction by segmentation of images for mobile robot navigation. *IEEE Int. Conf. on Robotics and Automation*, 1:95–100, 1996.

[41] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[42] James P Underwood, Andrew Hill, Thierry Peynot, and Steven J Scheding. Error modeling and calibration of exteroceptive sensors for accurate mapping applications. *J. of Field Robotics*, 27(1):2–20, 2010.

[43] Jurgen Vandorpe, Hendrik Van Brussel, and Hong Xu. Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2d range finder. *IEEE Int. Conf. on Robotics and Automation*, 1:901–908, 1996.

[44] Sosuke Yamao, Hiroshi Hidaka, Shigeyuki Odashima, Shan Jiang, and Yuichi Murase. Calibration of a rotating 2d lrf in unprepared environments by minimizing redundant

measurement errors. *IEEE Int. Conf. on Advanced Intelligent Mechatronics*, pages 172–177, 2017.

[45] Yadan Zeng, Heng Yu, Houde Dai, Shuang Song, Mingqiang Lin, Bo Sun, Wei Jiang, and Max Q-H Meng. An improved calibration method for a rotating 2d lidar system. *Sensors*, 18(2):497, 2018.

[46] Ji Zhang and Sanjiv Singh. LOAM: lidar odometry and mapping in real-time. *Robotics: Science and Systems*, 2:9, 2014.

[47] Shi Hui Zhang, Fang Wen Zhu, Zheng Peng Yuan, and Jin Bo Chen. The laser scanning system calibrations in unmanned yard. *Advanced Materials Research*, 718:1585–1590, 2013.

[48] Shi Zhao, Tien-Fu Lu, Ben Koch, and Alan Hurdsman. A simulation study of sensor data fusion using ukf for bucket wheel reclaimer localization. *IEEE Int. Conf. on Automation Science and Engineering*, pages 1192–1197, 2012.

[49] Shi Zhao, Tien-Fu Lu, Ben Koch, and Alan Hurdsman. Stockpile modelling using mobile laser scanner for quality grade control in stockpile management. *12th Int. Conf. on Control Automation Robotics & Vision (ICARCV)*, pages 811–816, 2012.

[50] Shi Zhao, Tien-Fu Lu, Ben Koch, and Alan Hurdsman. Dynamic modelling of 3d stockpile for life-cycle management through sparse range point clouds. *Int. J. of Mineral Processing*, 125:61–77, 2013.

[51] Shi Zhao, Tien-Fu Lu, Ben Koch, and Alan Hurdsman. 3d stockpile modelling and quality calculation for continuous stockpile management. *Int. J. of Mineral Processing*, 140:32–42, 2015.

[52] Robert Zlot and Michael Bosse. Efficient large-scale 3d mobile mapping and surface reconstruction of an underground mine. *Field and service robotics*, pages 479–493, 2014.